Jacques BOUQUET
Pierre MAYÉ

Électronique numérique

en 26 fiches



EXPRESS

DUNOD

Jacques BOUQUET
Professeur d'électronique à Arras

Pierre MAYÉ
Professeur agrégé de physique
Ingénieur en électronique
et électromécanique
Enseignant en BTS d'électronique
et d'électrotechnique à Arras

Électronique numérique en 26 fiches

DUNOD

ISBN 978-2-10-055548-2
ISSN 1778 4514

Table des matières

FICHE 1

Algèbre binaire

I Définitions

Une variable binaire ou logique, couramment nommée *bit*, ne prend que deux états notés 1 et 0, ce qui s'exprime par :

$$x = 0 \text{ si } x \neq 1 \quad \text{et} \quad x = 1 \text{ si } x \neq 0$$

Si on note la variable vraie et $\overline{x}$ (x barre) la variable complémentée, on a :

$$\text{quand } x = 0, \text{ alors } \overline{x} = 1 \text{ et quand } x = 1, \text{ alors } \overline{x} = 0$$

II Fonctions logiques

- **Fonctions logiques de base (Fig. 1.1)**

  Elles suffisent à l'écriture et la réalisation de toute fonction logique d'un nombre quelconque de variables.

Fonction	Symbole	Équation logique	Table de vérité
NON *(Inverter)* Complémentation	E — [1] — S	$S = \bar{E}$ (*S* égale *E* barre, ou *E* complémenté)	E / S : 0 / 1 ; 1 / 0
ET *(AND)* Produit logique	A, B — [&] — S	$S = A \cdot B$ (*S* égale *A* et *B*) $S = A \cap B$	A B / S : 0 0 / 0 ; 0 1 / 0 ; 1 0 / 0 ; 1 1 / 1
OU inclusif *(OR)* Somme logique	A, B — [≥ 1] — S	$S = A + B$ (*S* égale *A* ou *B*) $S = A \cup B$	A B / S : 0 0 / 0 ; 0 1 / 1 ; 1 0 / 1 ; 1 1 / 1

Figure 1.1 Fonctions logiques de base

- **Autres fonctions logiques courantes (Fig. 1.2)**

Fonction	Symbole	Équation logique	Table de vérité
NON-ET *(NAND)*	A & S B	$S = \overline{A \cdot B}$ (*S* égale *A* et *B*, barre, *A* et *B*, complémenté)	A B | S 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NON-OU *(NOR)*	A ≥ 1 S B	$S = \overline{A + B}$ (*S* égale *A* ou *B*, barre, *A* ou *B*, complémenté)	A B | S 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
OUI (*buffer*)	E 1 S	$S = E$ (*S* égale *E*)	E | S 0 | 0 1 | 1
OU exclusif *(XOR)*	A $= 1$ S B	$S = A \oplus B$ (*S* égale *A* ou exclusif *B*)	A B | S 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
IDENTITÉ (OU exclusif complémenté) *(XNOR)*	A $= 1$ S B	$S = \overline{A \oplus B}$ (*S* égale *A* ou exclusif complémenté *B*) ou $S = A \odot B$ (*S* égale *A* identique *B*)	A B | S 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Figure 1.2 Autres fonctions logiques courantes

III Propriétés des fonctions logiques de base

- **Règles de priorité**

Dans une équation logique la complémentation a la plus forte priorité, puis vient la fonction ET, et la fonction OU. L'utilisation de parenthèses dans les équations logiques permet de s'affranchir de ces priorités : ce sont les opérations à l'intérieur des parenthèses qui deviennent prioritaires.

- **Tableau des propriétés des fonctions logiques de base (Fig. 1.3)**

	Fonction OU	Fonction ET
Commutativité	$a + b = b + a$	$a \cdot b = b \cdot a$
Associativité	$a + (b + c) = (a + b) + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
Distributivité	$a \cdot (b + c) = a \cdot b + a \cdot c$	$a + (b \cdot c) = (a + b) \cdot (a + c)$
Idempotence	$a + a = a$	$a \cdot a = a$
Élément neutre	$a + 0 = a$	$a \cdot 1 = a$
Élément absorbant	$a + 1 = 1$	$a \cdot 0 = 0$
Complémentation	$a + \overline{a} = 1$	$a \cdot \overline{a} = 0$
Involution	$\overline{\overline{a}} = a$	

Figure 1.3 Propriétés des fonctions logiques de base

- **Règles de De Morgan**

Première règle : Le complément d'un produit de variables est égal à la somme de leurs compléments :

$$\overline{a \cdot b} = \overline{a} + \overline{b}$$

Deuxième règle : Le complément d'une somme de variables est égal au produit de leurs compléments :

$$\overline{a + b} = \overline{a} \cdot \overline{b}$$

- **Propriétés induites**

$$a + \overline{a} \cdot b = a + b \text{ car } \overline{\overline{a + \overline{a} \cdot b}} = \overline{\overline{a} \cdot (a + \overline{b})} = \overline{a \cdot \overline{a} + \overline{a} \cdot \overline{b}} = \overline{\overline{a} \cdot \overline{b}} = a + b$$

$$a + a \cdot b = a$$

$$(a + b) \cdot (a + c) = a + b \cdot c$$

$$a \cdot b + \overline{a} \cdot c = a \cdot b + \overline{a} \cdot c + b \cdot c$$

IV Définition d'une fonction logique quelconque

- **Fonction logique quelconque complètement définie**

Une fonction de n variables binaires est complètement définie si sa valeur est connue pour chacune des 2^n combinaisons possibles des variables. La table de véri-

té d'une telle fonction comportera donc 2^n lignes. Un exemple de table de vérité d'une fonction F de trois variables a, b et c est donné ci-dessous (Fig. 1.4).

a	*b*	*c*	*F*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Figure 1.4 Table de vérité d'une fonction complètement définie

- **Fonction logique quelconque partiellement définie**

Dans ce cas la valeur de la fonction n'est pas déterminée pour toutes les combinaisons des variables. Une valeur indéterminée est notée X ou Ø dans la table de vérité. Pour les cas indéterminés, on peut imposer une valeur 0 ou 1 à cette fonction, dans le but de faciliter sa synthèse. Un exemple d'une telle fonction est donné ci-après (Fig. 1.5).

g	*h*	*i*	*J*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	X
1	0	0	X
1	0	1	0
1	1	0	X
1	1	1	X

Figure 1.5 Table de vérité d'une fonction partiellement définie

- **Fonctions ou termes produits, mintermes**

Pour chacune des combinaisons des variables on définit une fonction égale au ET entre les variables vraies ou complémentées. Pour une fonction de trois variables a, b et c, il y a huit fonctions, ou termes produits, ce sont les mintermes :

$$P_0 = \overline{a} \cdot \overline{b} \cdot \overline{c}$$

$$P_1 = \overline{a} \cdot \overline{b} \cdot c$$

$$P_2 = \overline{a} \cdot b \cdot \overline{c}$$

$$P_3 = \overline{a} \cdot b \cdot c$$

$$P_4 = a \cdot \overline{b} \cdot \overline{c}$$

$$P_5 = a \cdot \overline{b} \cdot c$$

$$P_6 = a \cdot b \cdot \overline{c}$$

$$P_7 = a \cdot b \cdot c$$

- **Fonctions ou termes sommes, maxtermes**

De même, pour chacune des combinaisons des variables on définit une fonction égale au OU entre les variables vraies ou complémentées. Pour une fonction de trois variables a, b et c, il y a huit fonctions, ou termes sommes, ce sont les maxtermes :

$$S_0 = \overline{a} + \overline{b} + \overline{c}$$

$$S_1 = \overline{a} + \overline{b} + c$$

$$S_2 = \overline{a} + b + \overline{c}$$

$$S_3 = \overline{a} + b + c$$

$$S_4 = a + \overline{b} + \overline{c}$$

$$S_5 = a + \overline{b} + c$$

$$S_6 = a + b + \overline{c}$$

$$S_7 = a + b + c$$

- **Formes canoniques des fonctions logiques**

Une fonction logique peut s'écrire sous forme de somme de produits de variables vraies ou complémentées, ou sous forme de produit de sommes de variables vraies ou complémentées.
On obtient la première forme canonique en réunissant par des fonctions OU tous les termes produits, ou mintermes, pour lesquels la fonction vaut 1.
Ainsi, de la table de vérité (Fig. 1.4) on tire une équation logique donnant F :

$$F = \overline{a} \cdot \overline{b} \cdot c + \overline{a} \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot \overline{c}$$

La deuxième forme canonique s'obtient en réunissant par des fonctions ET tous les termes sommes, ou maxtermes, pour lesquels la fonction vaut 0.

Ainsi, de la table de vérité (Fig. 1.4) on tire une deuxième équation logique donnant F :

$$F(a+b+c)\cdot(a+\overline{b}+\overline{c})\cdot(a+b+\overline{c})\cdot(a+\overline{b}+c)\cdot(\overline{a}+\overline{b}+\overline{c})$$

Table de vérité et formes canoniques

1. Écrire la table de vérité d'une fonction logique F de trois variables a, b et c, telle que cette fonction soit égale à 1 si un nombre pair ou nul de variables est au niveau 1.
2. Tirer de cette table de vérité les deux formes canoniques de cette fonction.

Solution

1. La table de vérité s'obtient immédiatement :

a	b	c	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

2. La première forme canonique est obtenue en réunissant par des fonctions OU les mintermes pour lesquels $F = 1$:

$$F = \overline{a}\cdot\overline{b}\cdot\overline{c}+\overline{a}\cdot b\cdot c+a\cdot\overline{b}\cdot c+a\cdot b\cdot\overline{c}$$

La deuxième forme canonique est obtenue en réunissant par des fonctions ET les maxtermes pour lesquels $F = 0$:

$$F = (a+b+\overline{c})\cdot(a+\overline{b}+c)\cdot(\overline{a}+b+c)\cdot(\overline{a}+\overline{b}+\overline{c})$$

FICHE 2

Principes d'une minimisation

I Introduction

La minimisation consiste à simplifier l'expression d'une fonction logique dans le but d'optimiser le nombre de composants, ou portes, nécessaires à sa réalisation. L'expression obtenue est la forme minimale.

II Simplification analytique

Il convient d'abord de tirer de la table de vérité la forme canonique comportant le moins de termes. Pour réaliser une simplification analytique on utilise les propriétés des fonctions logiques (*cf.* fiche 1).
Présentons la méthode sur un exemple (Fig. 2.1).

a	b	c	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure 2.1 Exemple de table de vérité

L'équation tirée de la table de vérité s'écrit :

$$S = \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot c + a \cdot b \cdot \overline{c} + a \cdot b \cdot c$$

Comme $a \cdot b \cdot c = a \cdot b \cdot c + a \cdot b \cdot c + a \cdot b \cdot c$, nous pouvons transformer l'expression de la fonction :

$$S = (\overline{a} \cdot b \cdot c + a \cdot b \cdot c) + (a \cdot \overline{b} \cdot c + a \cdot b \cdot c) + (a \cdot b \cdot \overline{c} + a \cdot b \cdot c)$$

Après regroupement des termes et factorisation, nous obtenons :

$$S = b \cdot c \cdot (\overline{a} + a) + a \cdot c \cdot (\overline{b} + b) + a \cdot b \cdot (\overline{c} + c)$$

Or, $\overline{a} + a = 1$, $\overline{b} + b = 1$ et $\overline{c} + c = 1$.
Nous en déduisons une forme minimale :

$$S = b \cdot c + a \cdot c + a \cdot c$$

Cette méthode repose beaucoup sur l'intuition. Sa mise en œuvre devient difficile et fastidieuse lorsque le nombre de variables est grand. On lui préfère alors la méthode graphique qui utilise les tableaux de Karnaugh.

III Simplification au moyen des tableaux de Karnaugh

- **Principe**

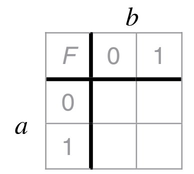
  Cette méthode de simplification consiste à mettre en évidence graphiquement les groupements de termes produits qui ne diffèrent que par l'état d'une seule variable d'entrée (termes adjacents).

- **Description des tableaux de Karnaugh**

  Un tableau de Karnaugh est une autre forme de la table de vérité. Il est organisé en colonnes et lignes dont les intersections donnent une case qui représente un des termes produits de la fonction. Pour une fonction de n variables, le tableau comportera 2^n cases. On écrit dans chaque case la valeur correspondante de la fonction : 0 ou 1, et si cette valeur est indéterminée, $\varnothing$ ou X.
  Les tableaux de Karnaugh sont bien adaptés pour la représentation de fonctions de quatre variables au plus.

- **Construction des tableaux de Karnaugh**

  Lignes et colonnes sont repérées par une combinaison des variables, sous forme littérale et par valeurs. La construction du tableau de Karnaugh est telle que les lignes et les colonnes successives sont repérées par des combinaisons adjacentes. Les tableaux de Karnaugh couramment utilisés concernent des fonctions de deux (Fig. 2.2), trois (Fig. 2.3), ou quatre variables (Fig. 2.4).



b (columns), *a* (rows)

F	0	1
0		
1		

Figure 2.2 Organisation du tableau de Karnaugh pour une fonction *F* de deux variables *a* et *b*

bc (columns), *a* (rows)

F	00	01	11	10
0				
1				

Figure 2.3 Organisation du tableau de Karnaugh pour une fonction *F* de trois variables *a*, *b* et *c*

cd (columns), *ab* (rows)

F	00	01	11	10
00				
10				
11				
01				

Figure 2.4 Organisation du tableau de Karnaugh pour une fonction *F* de quatre variables *a*, *b*, *c* et *d*

- **Passage de la table de vérité au tableau de Karnaugh**

Il suffit de reporter dans chaque case du tableau de Karnaugh la valeur de la variable de sortie correspondant à chaque combinaison des variables d'entrée (Fig. 2.5).

a	*b*	*c*	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

bc (columns), *a* (rows)

F	00	01	11	10
0	0	1	1	0
1	1	1	0	0

Figure 2.5 Transformation d'une table de vérité en tableau de Karnaugh

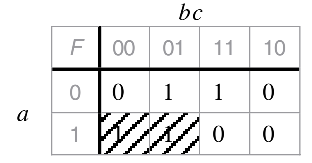
- **Utilisation des tableaux de Karnaugh**

  L'équation est obtenue en procédant à des groupements de cases adjacentes. Les règles à respecter sont :
  - le nombre de cases groupées doit être égal à une puissance de 2 ;
  - le nombre de cases entrant dans un groupement doit être le plus grand possible ;
  - une case peut être incluse dans plusieurs groupements ;
  - une case contenant un état indéterminé peut être incluse dans les groupements.

- **Exemple d'utilisation d'un tableau de Karnaugh**

  Les groupements mis en évidence (Fig. 2.6) donnent les deux termes produits $a \cdot \overline{b}$ et $\overline{a} \cdot b$. Une équation de F est donc :

$$F = a \cdot \overline{b} + \overline{a} \cdot b$$



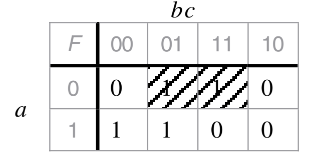


Figure 2.6 Définition des groupements

Tableau de Karnaugh d'une fonction de quatre variables

1. Écrire le tableau de Karnaugh d'une fonction R de 4 variables w, x, y et z définie par la table de vérité ci-contre.

2. En déduire une équation de R.

w	x	y	z	R
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Solution

1. On transpose la table de vérité dans un tableau de Karnaugh :

yz (colonnes), *wx* (lignes)

R	00	01	11	10
00	1	0	0	1
01	1	0	1	1
11	1	0	1	1
10	1	0	0	1

2. On fait apparaître deux groupements :

yz (colonnes), *wx* (lignes)

R	00	01	11	10
00	1	0	0	1
01	1	0	1	1
11	1	0	1	1
10	1	0	0	1

Premier groupement $x \cdot y$

yz (colonnes), *wx* (lignes)

R	00	01	11	10
00	1	0	0	1
01	1	0	1	1
11	1	0	1	1
10	1	0	0	1

Deuxième groupement : $\overline{z}$

L'équation de la fonction est obtenue par la réunion des deux groupements :

$$R = \overline{z} + x \cdot y$$

FICHE 3

Symboles graphiques normalisés

La norme NF C 03-212 ou CEI 617-12-1991 définit la représentation graphique des circuits logiques.

I Formation des symboles

- **Symbole élémentaire**

La représentation graphique d'un circuit logique est formée d'un cadre rectangulaire complété d'un symbole distinctif. Les entrées se trouvent à gauche et les sorties à droite (Fig. 3.1).

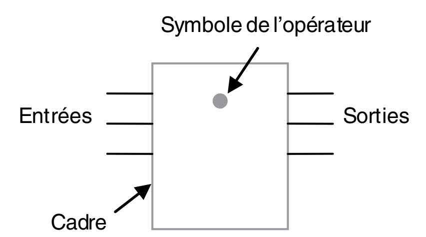


Figure 3.1 Symbole élémentaire

- **Association de symboles**

On peut associer plusieurs symboles en parallèle (Fig. 3.2) ou en cascade (Fig. 3.3).

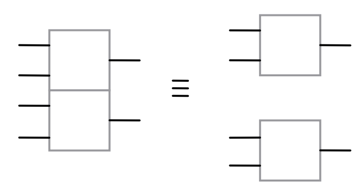


Figure 3.2 Associations de symboles en parallèle

Figure 3.3 Associations de symboles en cascade

- **Cadre des entrées communes**

Des cadres peuvent être surmontés d'un bloc des entrées communes où arrivent des signaux qui agissent sur chacun des opérateurs (Fig. 3.4).

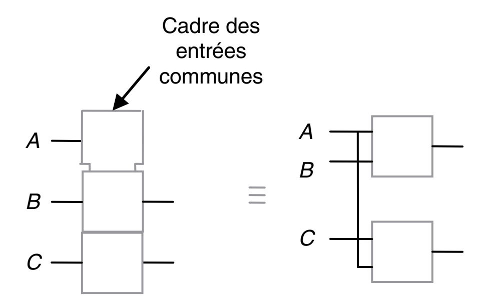


Figure 3.4 Cadre des entrées communes

- **Cadre des sorties communes**

Si la sortie d'un opérateur est affectée par l'ensemble des autres sorties, elle apparaît dans le cadre des sorties communes placé en-dessous des opérateurs concernés (Fig. 3.5).

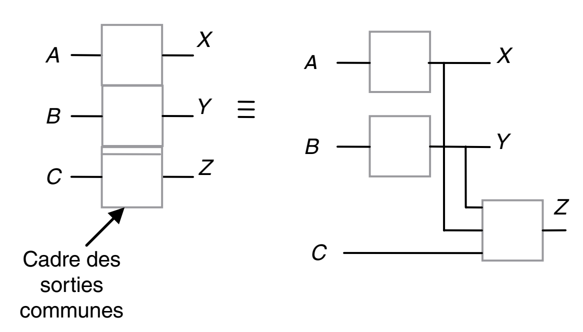


Figure 3.5 Cadre des sorties communes

- **Notation de dépendance**

Les entrées et les sorties associées à un chiffre sont liées par une relation de dépendance aux entrées et aux sorties portant ce même chiffre. Le type de dépendance est précisé par une lettre (Fig. 3.6).

Lettre	Type de dépendance	Lettre	Type de dépendance
G	ET	Z	Interconnexion
V	OU	A	Adresse
N	Négation	C	Commande
S	Mise à 1	EN	Validation
R	Mise à 0	M	Mode

Figure 3.6 Types de dépendance

II Symboles distinctifs

- **Symboles distinctifs des opérateurs**

Différents symboles sont placés dans les cadres pour définir les opérateurs (Fig. 3.7). Ils peuvent être associés, la fonction principale étant écrite en premier.

Symbole	Signification
1	OUI L'opérateur NON utilise ce symbole complété par le symbole de négation en sortie
&	ET
≥ 1	OU
$= 1$	OU exclusif
$\geq m$	Seuil logique
$= m$	m et seulement m
▷	Opérateur ayant un courant de sortie maximal plus élevé que les autres
⎍	Hystérésis
$2K$	Parité
$2K + 1$	Imparité
⎍	Monostable redéclenchable
1 ⎍	Monostable
G ⎍⎍	Astable

Figure 3.7 Symboles distinctifs des opérateurs

- **Symboles distinctifs des connexions**

Les entrées, sorties et autres connexions peuvent être associées à différents symboles (Fig. 3.8).

Symbole	Signification
	Négation logique en entrée
	Négation logique en sortie
	Polarité logique en entrée
	Polarité logique en sortie
	Entrée dynamique
	Entrée avec hystérésis
EN	Entrée de validation (*enable*)
	Sortie à courant amplifié
	Sortie à circuit ouvert de type L (collecteur ou drain ouvert)
	Sortie à circuit ouvert de type H
	Sortie trois états
D	Entrée d'une bascule D (même principe pour les autres bascules avec R, S, J et K)
+ m	Entrée de comptage
− m	Entrée de décomptage
CI	Entrée de retenue
CO	Sortie de retenue
CT = m	Entrée imposant un contenu (si $m = 0$, utiliser R)

Figure 3.8 Symboles distinctifs des connexions

L'indication directe de polarité logique consiste à donner directement la relation entre l'état logique interne et le niveau logique externe à un accès d'opérateur logique grâce à la présence ou à l'absence du symbole de polarité. Dans ce cas, le symbole de négation logique ne doit pas être utilisé.

La présence de l'indicateur de polarité logique signifie que le niveau bas L de la grandeur physique correspond à l'état logique 1. L'absence de cet indicateur signifie que le niveau haut H correspond à l'état 1.

- **Abréviations distinctives d'opérateurs**

Les opérateurs sont désignés par des abréviations (issues de mots anglais) dont nous donnons quelques exemples (Fig. 3.9).

Abréviation	Signification en anglais	Traduction en français
ALU	*Arithmetic logic unit*	Unité arithmétique et logique
CLK	*Clock*	Horloge
CPU	*Computer unit*	Microcalculateur
CTR	*Counter*	Compteur
CTRDIV	*Counter divider*	Compteur diviseur
DMX	*Demultiplexer*	Démultiplexeur
MEM	*Memory*	Mémoire
MUX	*Multiplexer*	Multiplexeur
PLA	*Programmable logic array*	Réseau logique programmable
REG	*Register*	Registre
UART	*Universal asynchronous receiver-transmitter*	Émetteur-récepteur asynchrone universel

Figure 3.9 Quelques abréviations d'opérateurs

Lecture de symboles

Donner la signification des différents symboles ci-dessous (Fig. 3.10).

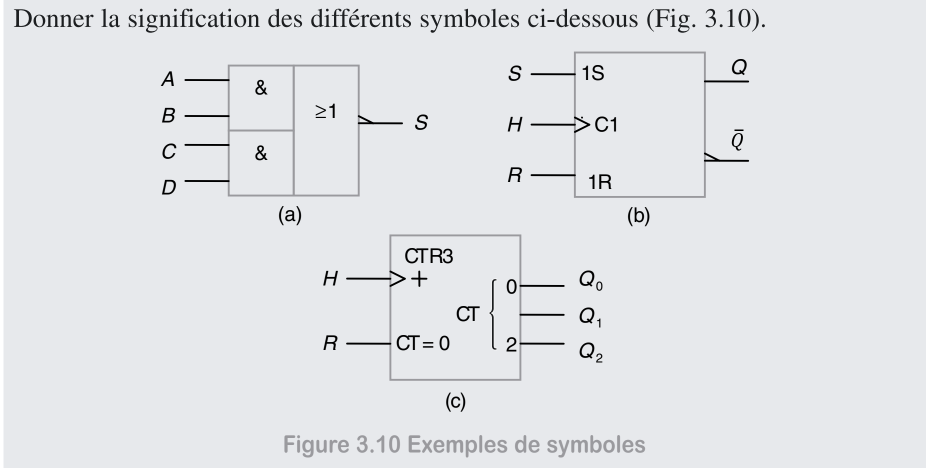


Figure 3.10 Exemples de symboles

Solution

Le symbole (a) est celui d'un circuit combinatoire qui correspond à l'équation logique $S = \overline{A \cdot B + C \cdot D}$.

Le symbole (b) est celui d'une bascule RS synchrone active sur les fronts montants de l'horloge *H*. Le triangle à l'intérieur du symbole en face de *H* indique que cette entrée est dynamique (active sur les fronts). En l'absence de symbole de polarité logique, c'est le front montant qui est actif. Le numéro 1 sur les entrées 1S et 1R montre que leur action est commandée (lettre C et numéro 1) par l'horloge.

Le symbole (c) est celui d'un compteur (CTR) à trois sorties en binaire naturel. Il possède une entrée de remise à zéro (CT = 0) qui force le contenu à 0 lorsque son état logique est 1.

FICHE 4

Introduction aux circuits logiques programmables

I Présentation

- **Définition**

Un circuit logique programmable, ou PLD (*Programmable Logic Device*) est un composant logique dont la fonction est définie, puis fixée par l'utilisateur lors d'une opération appelée programmation qui peut être définitive ou temporaire.
Ces composants existent sous diverses appellations commerciales qui recouvrent des technologies différentes ou font référence à certains fabricants.

- **Structure de base des circuits logiques programmables**

La structure d'un circuit logique programmable peut découler de la forme « somme logique de produits logiques » que peut prendre une fonction logique. Elle se compose dans ce cas de quatre couches :

– la première reçoit en entrée des variables logiques et fournit en sortie ces mêmes variables, vraies et complémentées ;
– la deuxième réalise les produits logiques qui apparaissent dans l'expression de la fonction logique : c'est le réseau ou la matrice des ET d'entrée ;
– la troisième réalise la somme logique de ces produits logiques : c'est le réseau ou la matrice des OU de sortie ;
– la quatrième couche constitue la structure de sortie. Sous sa forme la plus complète elle fournit la valeur vraie et complémentée de la fonction logique, permet la logique trois états, et introduit une fonction mémorisation avec une bascule, de type D en général.

Les variables issues de la structure de sortie sont ajoutées aux variables d'entrée de la première couche, ce qui permet de réaliser des fonctions logiques combinatoires qui comportent de nombreuses variables d'entrée, et des fonctions logiques séquentielles.

- **Convention de représentation**

Les entrées des opérateurs sont organisées en bus. La représentation utilise habituellement une croix pour indiquer qu'une connexion entre une entrée d'opérateur et une sortie d'un autre opérateur, ou une variable, est programmable. Un point indique une connexion non programmable, permanente. La programmation consiste à valider ou non les connexions programmables, au moyen d'un appareil : le programmateur. Lorsqu'il ne doit pas avoir connexion d'une entrée à une variable, l'entrée est portée au niveau logique neutre de l'opérateur.
Les schémas suivants (Fig. 4.1) représentent des opérateurs OU à quatre entrées. Chacune des entrées est connectée à l'une des quatre variables *a*, *b*, *c*, *d*.

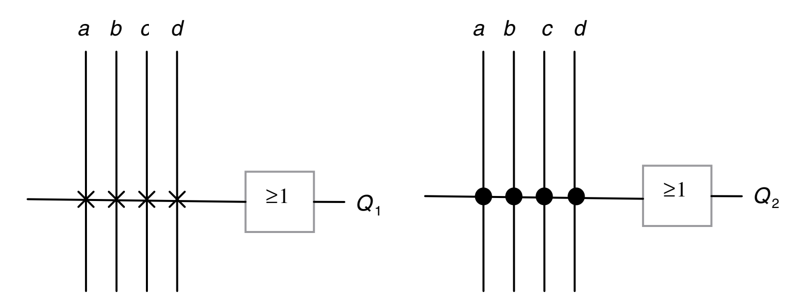


Figure 4.1 Convention de représentation

II Différentes structures

- **Première structure : la mémoire à lecture seule (Fig. 4.2)**

Dans cette structure le réseau des ET d'entrée est fixe et joue le rôle de décodeur d'adresse, alors que le réseau des OU de sortie est programmable : c'est une mémoire à lecture seule, programmable.
Les bits *a* et *b* forment les adresses, les bits S_1, S_2, S_3 et S_4 les données. *CE* est un signal de validation des sorties.
Quand elle n'est pas programmée, cette mémoire satisfait les équations suivantes :

$$Q_1 = \overline{a} \cdot \overline{b}$$

$$Q_2 = \overline{a} \cdot b$$

$$Q_3 = a \cdot \overline{b}$$

$$Q_4 = a \cdot b$$

$$S_1 = S_2 = S_3 = S_4 = Q_1 + Q_2 + Q_3 + Q_4 \text{ , quand } CE = 0$$

Les sorties S_1, S_2, S_3 et S_4 sont dans l'état haute impédance quand $CE = 1$.

Selon la technologie, ce type de mémoire est programmable une seule fois, car non effaçable (PROM en technologie bipolaire, programmation par destruction de fusibles), ou reprogrammable car effaçable, soit par exposition à un rayonnement ultraviolet (technologie MOS EPROM), soit électriquement (technologie MOS EEPROM).

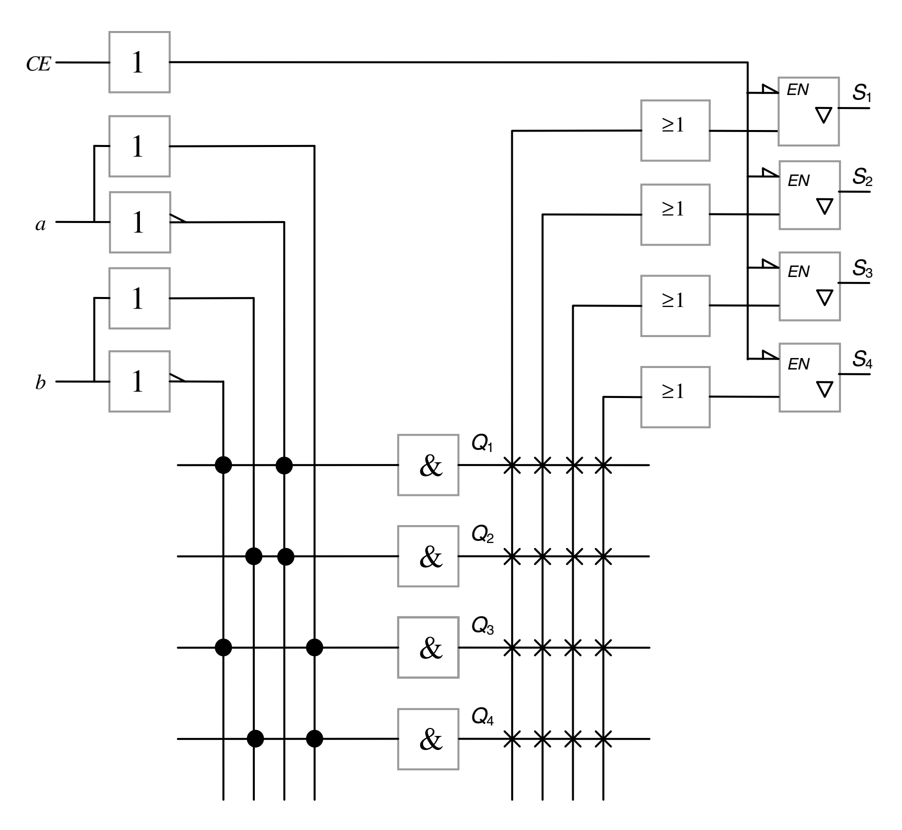


Figure 4.2 Structure d'une mémoire à lecture seule

- **Deuxième structure : le PAL (*Programmable Array Logic*) (Fig. 4.3)**

Le réseau des ET d'entrée est programmable, le réseau des OU de sortie est fixe. L'équation des variables de sortie S_1, S_2, S_3 et S_4 a la forme d'une somme logique de produits logiques, ces produits pouvant être choisis par programmation. Quand il n'est pas programmé, ce PAL satisfait les équations suivantes :

$$Q_1 = Q_2 = Q_3 = Q_4 = a \cdot b \cdot \overline{a} \cdot \overline{b}$$

$$S_1 = Q_1 + Q_2$$

$$S_2 = Q_2 + Q_3$$

$$S_3 = Q_3 + Q_4$$

$$S_4 = Q_1 + Q_4$$

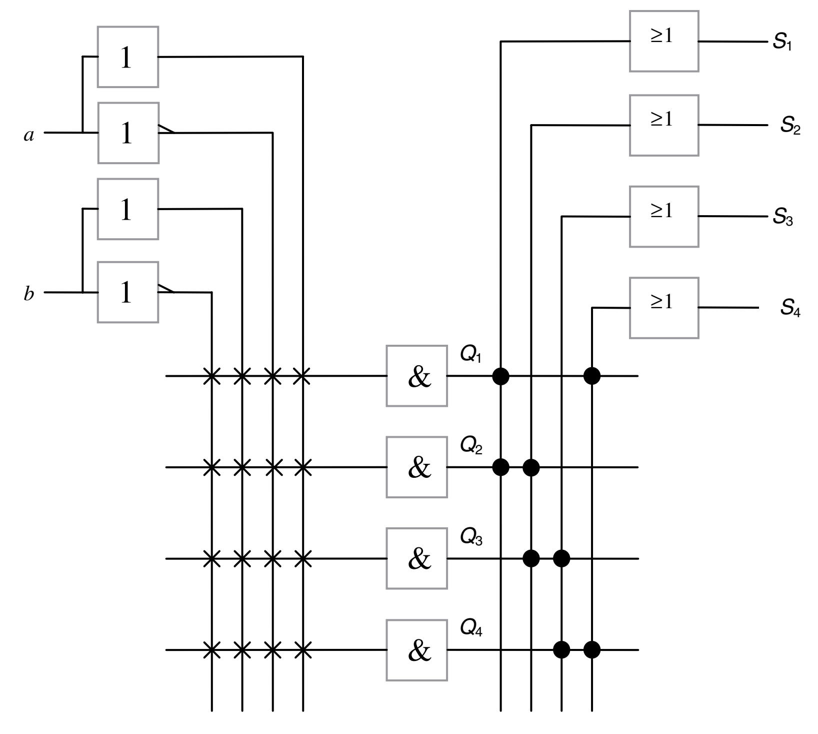


Figure 4.3 Structure d'un PAL

- **Troisième structure : le PLA (*Programmable Logic Array*) (Fig. 4.4)**

Le réseau des ET d'entrée et le réseau des OU de sortie sont programmables. Quand il n'est pas programmé, ce PLA satisfait les équations suivantes :

$$Q_1 = Q_2 = Q_3 = Q_4 = a \cdot b \cdot \overline{a} \cdot \overline{b}$$

$$S_1 = S_2 = S_3 = S_4 = Q_1 + Q_2 + Q_3 + Q_4$$

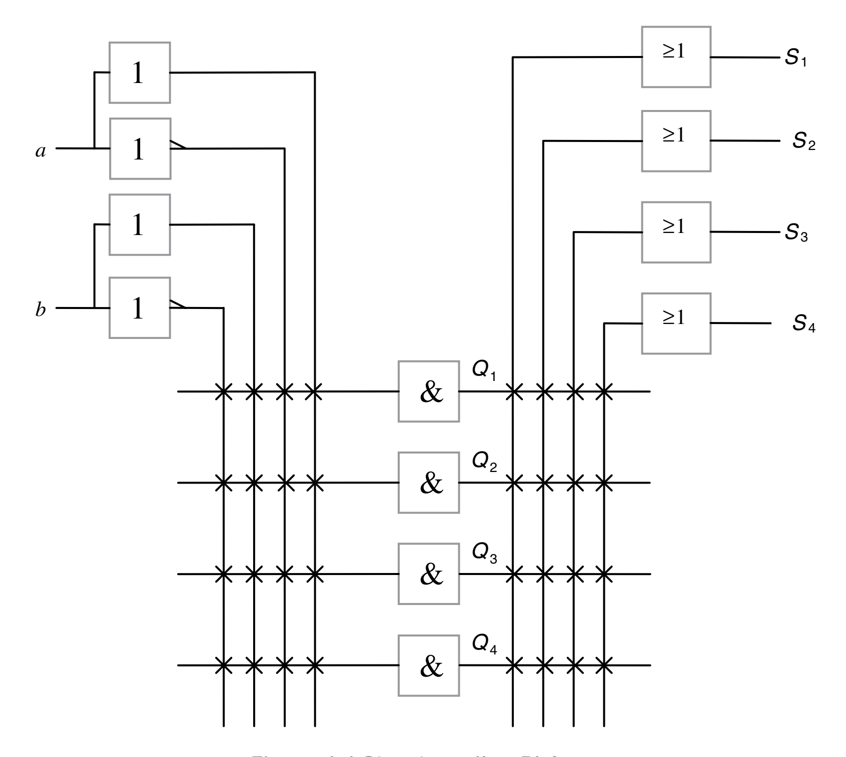


Figure 4.4 Structure d'un PLA

- **Structure de la cellule de sortie programmable (OLMC) d'un PAL (Fig. 4.5)**

La cellule de sortie est configurable par l'utilisateur. Les possibilités sont :

– la sortie reproduit celle de l'opérateur OU ;
– la sortie est complémentaire de l'opérateur OU ;
– la sortie est celle de l'opérateur OU, mémorisée par une bascule D ;
– la sortie est complémentaire de l'opérateur OU, mémorisée par une bascule D ;
– la sortie est en haute impédance, la broche correspondante peut être utilisée en entrée, qui s'intègre dans le réseau d'entrée.

La cellule de sortie est configurable par l'utilisateur. Les possibilités sont :

– la sortie reproduit celle de l'opérateur OU ;
– la sortie est complémentaire de l'opérateur OU ;
– la sortie est celle de l'opérateur OU, mémorisée par une bascule D ;
– la sortie est complémentaire de l'opérateur OU, mémorisée par une bascule D ;
– la sortie est en haute impédance, la broche correspondante peut être utilisée en entrée, qui s'intègre dans le réseau d'entrée.

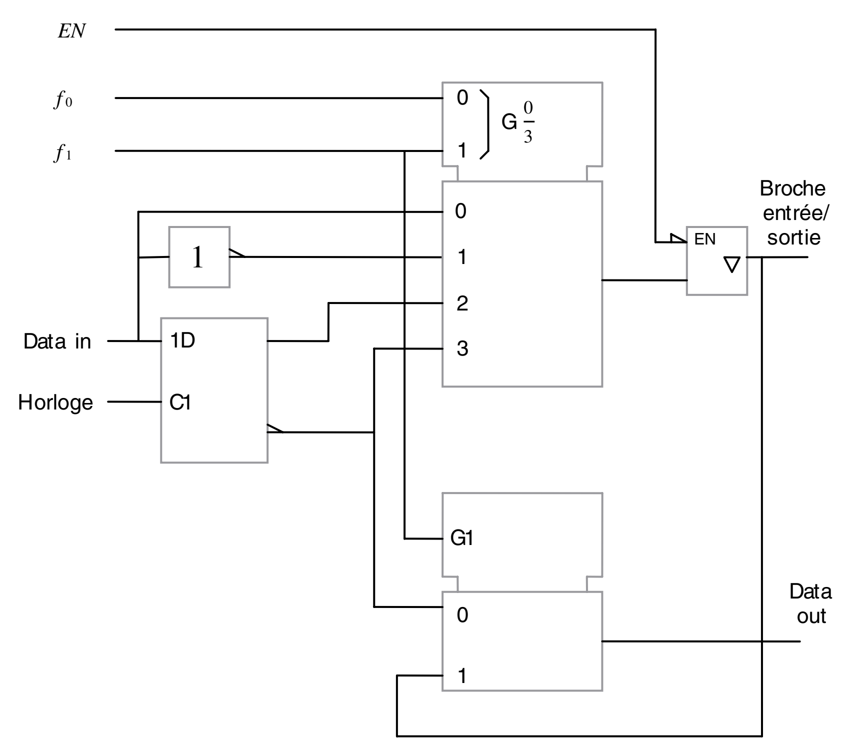


Figure 4.5 Structure de la cellule de sortie programmable (OLMC)

Choix d'un circuit logique programmable

On souhaite utiliser un circuit logique programmable pour réaliser un circuit séquentiel possédant douze entrées et cinq sorties et un circuit combinatoire ayant trois entrées et une sortie.
On dispose des trois circuits suivants :
– PAL 16R8,
– GAL 16V8,
– GAL 22V10.
Rechercher et consulter sur Internet les notices de ces trois composants pour répondre aux questions suivantes.
1. Quel circuit peut convenir pour l'application envisagée ? Justifier.
2. Quels sont les avantages des GAL par rapport aux PAL ?
3. Que signifient les chiffres et les lettres qui forment la dénomination de ces trois circuits ?

Solution

1. Le PAL 16R8 ne convient pas car il n'a pas de sorties combinatoires. Le GAL 16V8 ne peut pas non plus être utilisé ici car il ne dispose que de huit entrées et de huit entrées-sorties, ce qui est insuffisant pour l'application envisagée qui comporte un total de quinze entrées et six sorties.
Le GAL 22V10 convient car il dispose de douze entrées et de dix entrées-sorties, ce qui suffit pour notre cas.
2. Les PAL ne peuvent être programmés qu'une seule fois alors que les GAL sont reprogrammables autant que l'on veut. De plus, la consommation des GAL est plus faible.
3. Les deux premiers chiffres indiquent le nombre total de broches d'entrée ou d'entrée-sortie.
La lettre précise le type de sortie : R (*registred*) correspond à des sorties séquentielles et V (*versatile*) à des sorties universelles, c'est-à-dire soit combinatoires, soit séquentielles.
Les deux derniers chiffres donnent le nombre de broches configurables en sorties.
Par exemple, le GAL 22V10 comporte 22 broches d'entrée ou d'entrée-sortie et 10 broches configurables en sorties, ce qui donne 22 – 10 = 12 entrées pures.

FICHE 5

Circuits logiques programmables en logique combinatoire

I Introduction

Trois types de circuits permettent de coder des fonctions combinatoires :

– les multiplexeurs,
– les mémoires,
– les réseaux logiques programmables.

II Codage par multiplexeur

Un multiplexeur est un circuit qui possède n entrées d'adresse, 2^n entrées d'information, ou de donnée, et une sortie. Un multiplexeur à deux entrées d'adresse et donc quatre entrées de donnée (Fig. 5.1) est décrit par l'équation logique suivante :

$$S = \overline{A_1} \cdot \overline{A_0} \cdot I_0 + \overline{A_1} \cdot A_0 \cdot I_1 + A_1 \cdot \overline{A_0} \cdot I_2 + A_1 \cdot A_0 \cdot I_3$$

où les A_i sont les adresses, les I_j sont les données et S est la sortie.

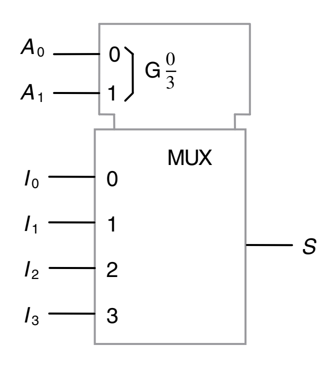


Figure 5.1 Multiplexeur

Les niveaux logiques appliqués sur les entrées I_0, I_1, I_2 et I_3 détermineront la fonction logique S réalisée entre les variables A_0 et A_1. Il y a 16 fonctions réalisables (Fig. 5.2)

I_3	I_2	I_1	I_0	Fonction S
0	0	0	0	0
0	0	0	1	$A_1 \cdot A_0$
0	0	1	0	$A_1 \cdot \overline{A_0}$
0	0	1	1	A_1
0	1	0	0	$\overline{A_1} \cdot A_0$
0	1	0	1	A_0
0	1	1	0	$\overline{A_1} \cdot A_0 + A_1 \cdot \overline{A_0} = A_1 \oplus A_0$
0	1	1	1	$A_1 + A_0$
1	0	0	0	$\overline{A_1} \cdot \overline{A_0}$
1	0	0	1	$\overline{A_1} \cdot \overline{A_0} + A_1 \cdot A_0 = A_1 \odot A_0$
1	0	1	0	$\overline{A_0}$
1	0	1	1	$A_1 + \overline{A_0}$
1	1	0	0	$\overline{A_1}$
1	1	0	1	$\overline{A_1} + A_0$
1	1	1	0	$\overline{A_1} + \overline{A_0}$
1	1	1	1	1

Figure 5.2 Fonctions réalisables

III Codage par mémoire

Une mémoire (Fig. 5.3) est équivalente à une table de correspondance : à un mot d'adresse en entrée on fait correspondre un mot de donnée en sortie. On peut donc mémoriser des tables de vérité et réaliser ainsi des fonctions logiques quelconques. Les bits du mot d'adresse seront assimilés aux variables d'entrée d'une fonction (ou circuit) logique et chacun des bits du mot de donnée donnera la valeur correspondante de la variable de sortie d'une fonction (ou circuit) logique (Fig. 5.4).

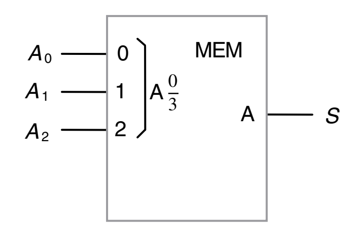


Figure 5.3 Mémoire

A_2	A_1	A_0	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$S = \overline{A_2} \cdot \overline{A_1} \cdot A_0 + \overline{A_2} \cdot A_1 \cdot \overline{A_0} + A_2 \cdot \overline{A_1} \cdot \overline{A_0}$$

Figure 5.4 Table de vérité

IV Codage par réseau logique

Il est basé sur la forme canonique « somme logique de produits logiques ». Un réseau logique réalise des produits logiques au moyen d'un réseau d'interconnexion programmable. Ces produits logiques constituent les entrées d'un opérateur OU qui exécute une somme logique. (Fig. 5.5)

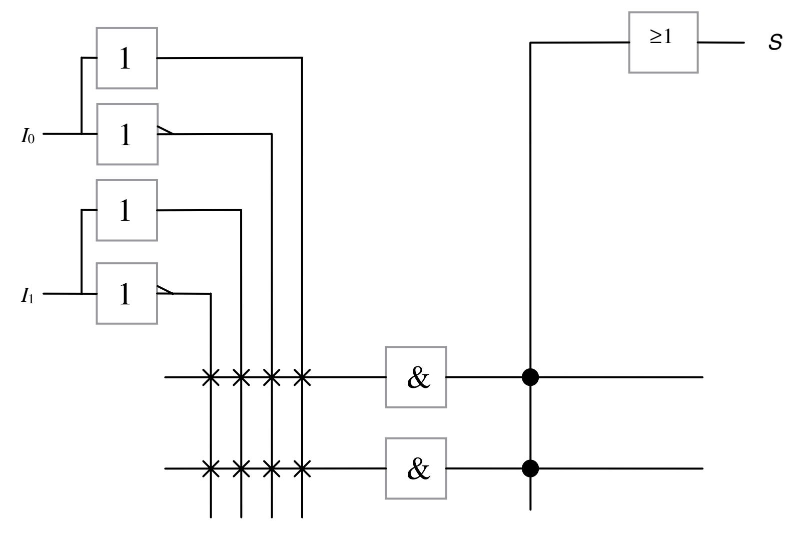


Figure 5.5 Réseau logique

L'équation logique d'un réseau à deux entrées I_0, I_1 et une sortie S est, lorsqu'il n'est pas encore programmé :

$$S = I_0 \cdot \overline{I_0} \cdot I_1 \cdot \overline{I_1} + I_0 \cdot \overline{I_0} \cdot I_1 \cdot \overline{I_1}$$

La programmation consiste à éliminer des connexions dans le réseau d'interconnexion de manière à réaliser la fonction souhaitée.

Par exemple, en supprimant la connexion de $\overline{I_0}$ et de I_1 du premier terme, et en outre la connexion de I_0 et de $\overline{I_1}$ du deuxième terme, on obtient la fonction OU exclusif :

$$S = I_0 \cdot \overline{I_1} + \overline{I_0} \cdot I_1 = I_0 \oplus I_1$$

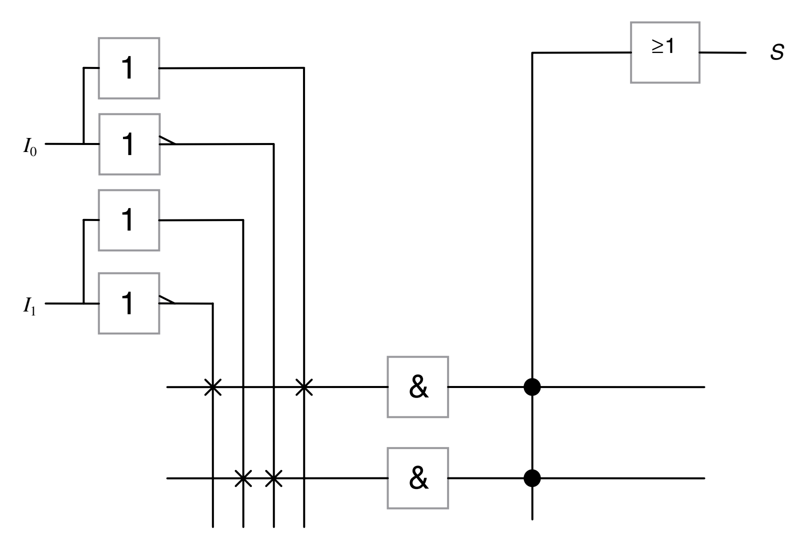


Figure 5.6 Élimination des connexions

Réseau logique

Indiquer la programmation à effectuer sur un réseau logique à trois entrées A, B, C et deux sorties S_1 et S_2 pour obtenir l'équation logique :

$$S_1 = A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C} + A \cdot \overline{B} \cdot C$$

Pour simplifier le schéma, les inverseurs permettant de passer de A, B, C à $\overline{A}$, $\overline{B}$, $\overline{C}$ ne seront pas représentés.

Solution

On élimine les connexions nécessaires pour d'obtenir la fonction désirée (Fig. 5.7).

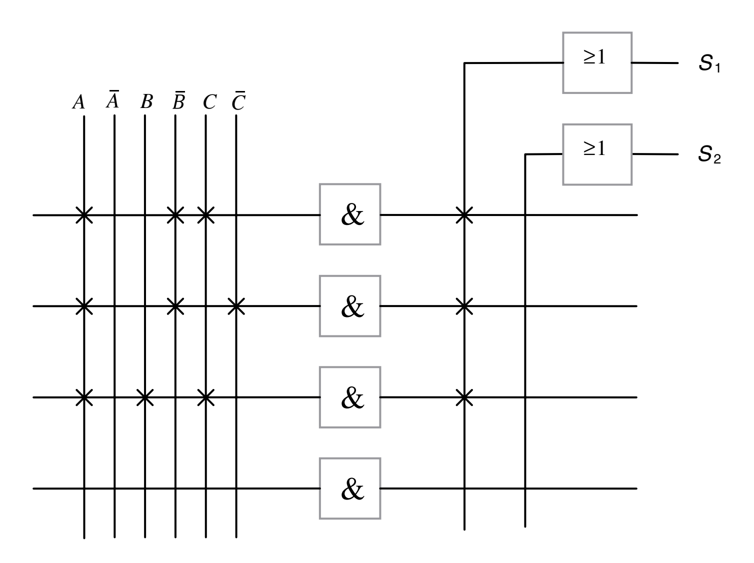


Figure 5.7 Réseau logique

FICHE 6

Description VHDL d'un circuit combinatoire

I Présentation

Le langage de description VHDL (*Very hight speed integrated circuit Hardware Description Language*) est un langage de description du comportement d'un circuit logique. C'est un langage de haut niveau : il fait abstraction de l'objet pour lequel il est utilisé. Le langage VHDL est standardisé depuis 1987 par la norme IEEE1076 de l'*Institute of Electrical and Electronics Engineers*.
Le langage VHDL est utilisé en particulier pour développer les circuits logiques programmables. Un outil de développement permet la programmation d'un circuit logique programmable à partir d'un fichier VHDL.

II Structure d'une description VHDL

La description VHDL d'un circuit combinatoire est composée de deux parties :

– l'entité (*entity*) qui permet de définir les entrées et les sorties du composant, leurs noms et leurs types ;

– l'architecture (*architecture*) qui permet de définir le comportement du circuit.

Prenons l'exemple le plus simple qui soit : la description d'un inverseur :

```
library ieee;
use ieee.std_logic_1164.all;
entity INVERSEUR is
port (
E: in std_logic; -- entree
S: out std_logic ); -- sortie
end INVERSEUR;
architecture ARCH_INVERSEUR of INVERSEUR is
begin
S <= not E;
end ARCH_INVERSEUR;
```

La description commence par la déclaration des bibliothèques : le mot `library` indique l'utilisation d'une bibliothèque et le mot `use` permet l'accès aux fonctions qu'elle contient.
La définition de l'entité est annoncée par le mot `entity`. L'utilisateur donne un nom à l'entité : nous avons choisi ici INVERSEUR. Le mot `port` permet de définir le nom, le mode et le type des signaux des entrées et des sorties.
À côté des définitions de l'entrée et de la sortie, nous avons placé un commentaire, annoncé par un double tiret (`- -`).
Les quatre modes possibles sont :

– `in` : entrée ;
– `out` : sortie ;
– `inout` : entrée-sortie ;
– `buffer` : sortie servant d'entrée interne.

Parmi les nombreux types possibles, citons :

– `bit` : signal binaire (0 ou 1) ;
– `bit_vector` : bus ;
– `std_logic` : variable logique standard qui, en dehors des valeurs 0 et 1, peut prendre un état haute impédance (Z) et un état indéterminé (`-`) ;
– `std_logic_vector` : le bus correspondant.

L'utilisation de ces deux derniers types, présents dans la bibliothèque standard IEEE doit être signalée au début par :

```
library ieee;
use ieee.std_logic_1164.all;
```

Quand la description VHDL est destinée à l'implantation dans un circuit logique programmable, un logiciel place les différents blocs qui correspondent à la structure. L'affectation des broches d'entrée et de sortie peut être automatique ou peut se faire manuellement. Dans ce cas, les numéros sont précisés dans l'entité.
La définition de l'architecture est annoncée par le mot `architecture`. Le symbole <= correspond à une affectation inconditionnelle : la grandeur S prend la valeur de not E. Le symbole `not` correspond à l'opérateur logique complément.
Les symboles des opérateurs logiques élémentaires sont assez naturels (Fig. 6.1).

Symbole	Opérateur
and	Et
or	Ou
nand	Non et
nor	Non ou
xor	Ou exclusif

Figure 6.1 Opérateurs logiques

Remarque. Le langage VHDL ne fait pas de distinction entre majuscules et minuscules. Nous avons choisi d'écrire en minuscules les mots du langage et en majuscules les noms attribués par l'utilisateur, mais c'est un choix parmi d'autres.

Décodage d'adresse

Un système assez ancien utilise un décodeur d'adresse en logique câblée. Afin de le moderniser, il est décidé d'intégrer dans un circuit logique programmable complexe (CPLD) l'ensemble de la logique du système. Le développement du circuit est effectué en langage VHDL.

L'étude demandée porte uniquement sur les deux sorties $\overline{\text{DTACK_LENT}}$ et $\overline{\text{VPA}}$ situées sur le schéma décodage d'adresse (Fig. 6.2).

1. Compléter le bloc fonctionnel ci-dessous (Fig. 6.3) en faisant apparaître les signaux d'entrée nécessaires pour obtenir les sorties.

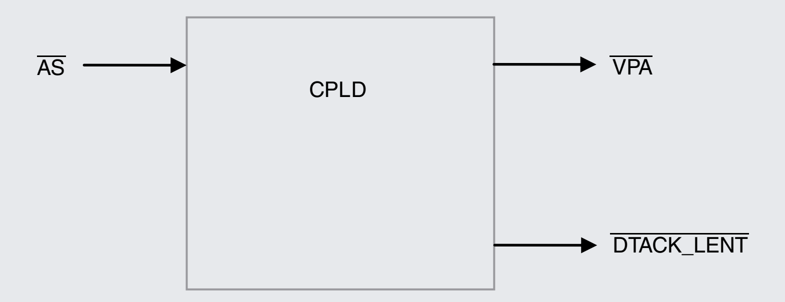


Figure 6.3 Bloc fonctionnel

2. Établir l'équation logique des sortie $\overline{\text{VPA}}$ et $\overline{\text{DTACK_LENT}}$.

3. Compléter le fichier VHDL et le schéma du composant (Fig. 6.4) permettant la programmation et l'affectation des signaux aux différentes broches.

Fichier VHDL :

```
library IEEE;
use IEEE.std_logic_1164.all;
entity decodeur is port (
    fc0,     .     .     .    :in std_logic;
    rw, lds:in std_logic;
    s1:out std_logic;
    .     .     .     .     .    :out std_logic;
attribute pin_numbers of decodeur : entity is
    " fc0:2    fc1: .  .   fc2:  .  .    as :9   a23 :  .  .  " &
    " dtack_lent:42       vpa:  .  .  " &
```

```
    " lds:20    rw:21      s1:24   " ;
end decodeur;
architecture arch_decodeur of decodeur is
begin
    vpa  . . . . . . . . . . . . . . .
    dtack_lent  . . . . . . . . . . . . . . .
    s1 <=  rw  or lds;
end arch_decodeur;
```

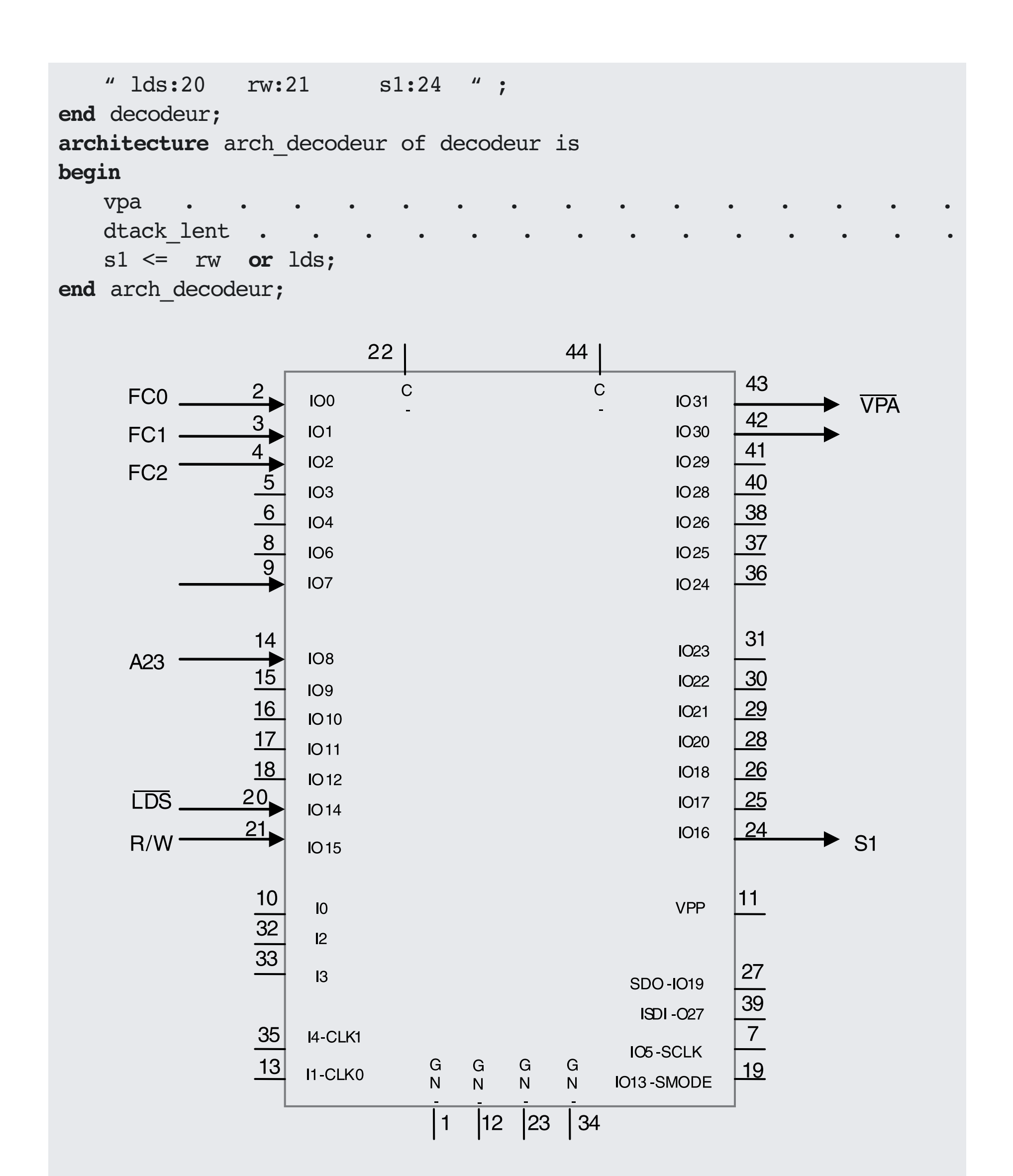


Figure 6.4 Schéma du CPLD

4. Quels sont les avantages et les inconvénients procurés par l'utilisation de circuits logiques programmables ?

Solution

1. Le bloc fonctionnel est complété par les entrées A23, FC0, FC1 et FC2 qui interviennent dans les expressions des sorties (Fig. 6.5).

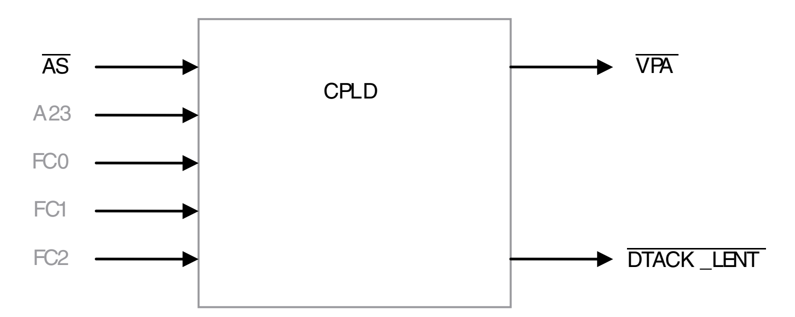


Figure 6.5 Bloc fonctionnel complété

2. Les équations logiques des sorties se lisent directement sur le schéma du décodeur d'adresse :

$$\overline{VPA} = \overline{A23}.\overline{AS.FC0.FC1.FC2}$$

$$\overline{DTACK_LENT} = \overline{\overline{AS}.\overline{A23}} = AS + A23$$

3. Le fichier VHDL est complété par l'indication des entrées Fc1, Fc2 et $\overline{AS}$ et de la sortie $\overline{VPA}$, par l'attribution des broches manquantes et par la définition des expressions de $\overline{VPA}$ et de DTACK_LENT correspondant aux équations logiques :

```
library IEEE;
use IEEE.std_logic_1164.all;
entity decodeur is port (
    fc0, fc1,fc2, as:in std_logic;
    rw, lds:in std_logic;
    s1:out std_logic;
    vpa:out std_logic;
attribute pin_numbers of decodeur : entity is
    " fc0:2   fc1:3    fc2:4    as:9    a23:14 "&
    " dtack_lent:42  vpa:43 "&
    " lds:20   rw:21   s1:24 ";
end decodeur;
architecture arch_decodeur of decodeur is
begin
    vpa <= (not a23) and not (as and fc0 and fc1 and fc2)
    dtack_lent <= as or a23
    s1 <=  rw  or lds;
end arch_decodeur;
```

Remarquons que cet énoncé ne fait pas le même choix que celui que nous avons adopté dans l'exemple quant à l'utilisation des majuscules et des minuscules : nous avons signalé que ce choix est arbitraire pour le langage VHDL.
Le schéma du composant est complété par l'entrée $\overline{\text{AS}}$ et la sortie $\overline{\text{DTACK_LENT}}$ (Fig. 6.6).

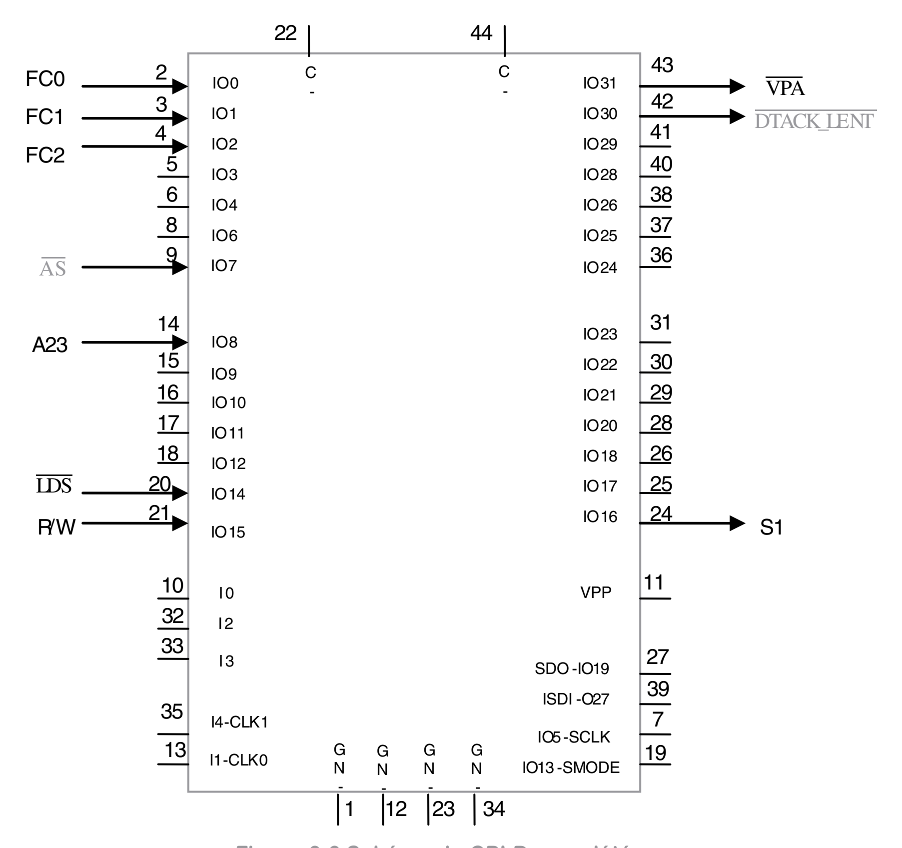


Figure 6.6 Schéma du CPLD complété

4. Les avantages des circuits logiques programmables sont :

– l'intégration d'une quantité importante de cellules logiques programmables,
– la vitesse de traitement.

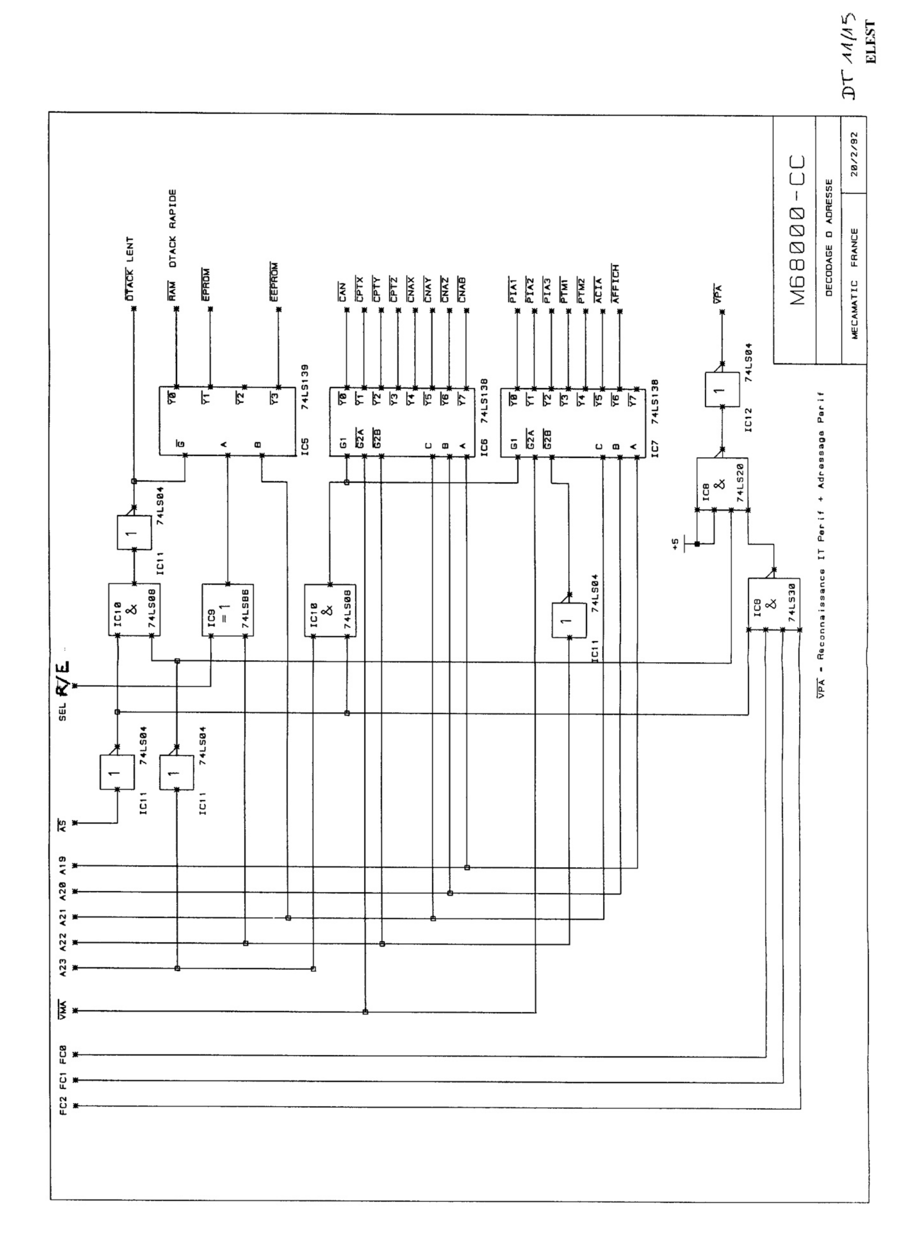


Figure 6.2 Décodeur d'adresse

FICHE 7

Systèmes logiques séquentiels

I Présentation

- **Définition**

Dans un système logique séquentiel une notion de mémoire intervient. L'état actuel du système dépend non seulement de l'état actuel des entrées, mais aussi de l'état antérieur dans lequel se trouvait le système, état matérialisé par des variables internes.

- **Système logique séquentiel asynchrone**

Un système logique séquentiel est dit à fonctionnement asynchrone si une modification de ses entrées a une conséquence directe sur ses sorties, après un retard dû au temps de propagation des opérateurs utilisés (Fig. 7.1).

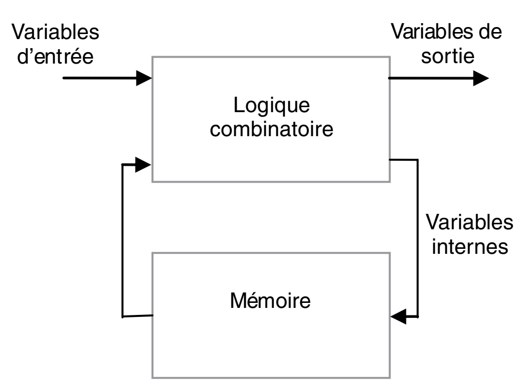


Figure 7.1 Système logique séquentiel asynchrone

- **Système logique séquentiel synchrone**

Dans le cas d'un système synchrone une modification des entrées n'est répercutée sur les sorties qu'après une validation par un signal de synchronisation, le signal d'horloge (Fig. 7.2).

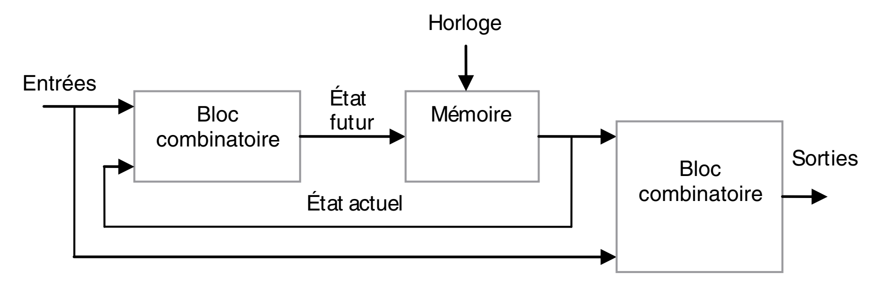


Figure 7.2 Système logique séquentiel synchrone

- **Modes de commande du système**

Les entrées du système peuvent être commandées soit par des niveaux logiques, soit par des impulsions, c'est-à-dire le passage momentané d'un niveau à un autre. Lors de la commande par niveaux, c'est la combinaison des niveaux logiques sur les entrées qui détermine l'évolution du système : s'il y a n entrées, il y aura 2^n possibilités. Lors de la commande par impulsion, comme il ne peut pas y avoir par définition deux impulsions simultanées, il n'y a que n possibilités.

II Machines à états

- **Définition**

Une machine à états (FSM pour *Finite State Machine*) est un système séquentiel qui peut se trouver dans un nombre fini d'états.

- **Structure**

L'architecture générale est celle d'un système séquentiel synchrone. La mémoire, appelée registre d'état, contient l'état actuel du système. Elle est constituée de n bascules synchrones. Le nombre maximal d'états possibles est 2^n.

- **Machine de Moore**

Dans une machine de Moore (Fig. 7.3), l'état des sorties est déterminé à partir de l'état actuel de la machine, résultant de l'état actuel des entrées et de l'état antérieur de la machine. C'est le vecteur d'état, constitué de variables internes, et mémorisé, qui permet de prendre en compte l'état antérieur de la machine.

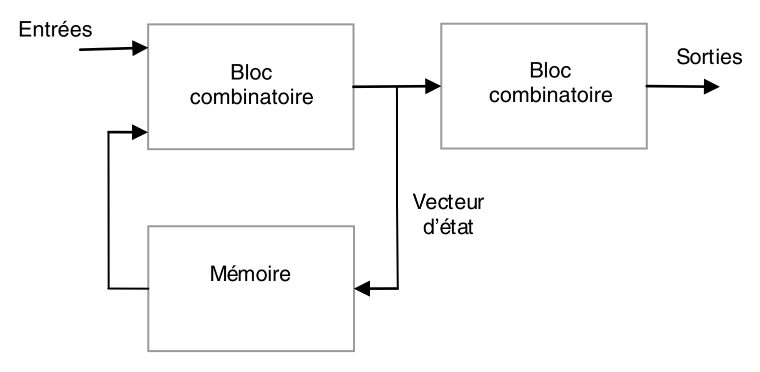


Figure 7.3 Machine de Moore

- **Machine de Mealy**

Dans une machine de Mealy (Fig. 7.4), l'état des sorties est déterminé non seulement à partir de l'état actuel de la machine, matérialisé par le vecteur d'état, mais aussi de l'état actuel des entrées.

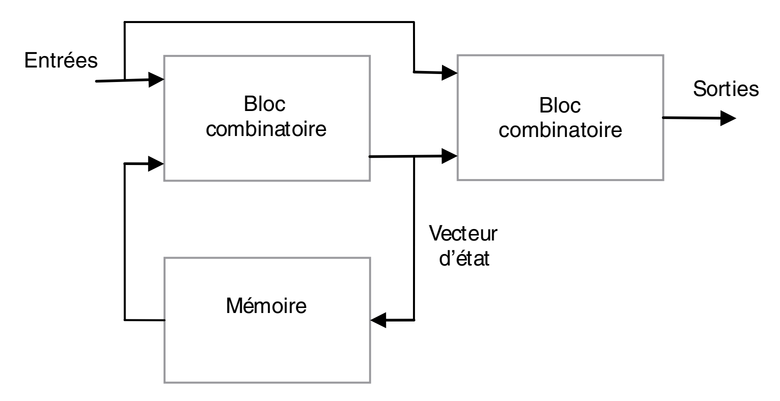


Figure 7.4 Machine de Mealy

III Diagramme d'état

- **Notions de base**

Un diagramme d'état est l'outil de description privilégié d'une machine à états. Il utilise deux symboles graphiques (Fig. 7.5) :

– les états (contenus possibles du registre d'état) sont représentés par des cercles ;
– les transitions (possibilités de passage d'un état à l'autre) par des arcs orientés.

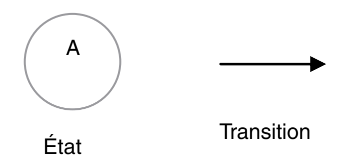


Figure 7.5 États et transitions

Une transition peut être inconditionnelle, c'est-à-dire que si le système est dans l'état source considéré, la transition se produit lors du front actif d'horloge suivant. On peut donner comme exemple un système possédant deux états A et B et qui change d'état à chaque front actif d'horloge (Fig. 7.6).

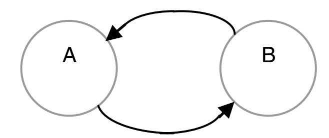


Figure 7.6 Transitions inconditionnelles

Le plus souvent, une transition est conditionnelle, c'est-à-dire que quand le système est dans l'état source, la transition se produit lors du front actif d'horloge suivant si une condition sur les entrées est vérifiée. Prenons un exemple (Fig. 7.7). À partir d'un état A, une transition peut conduire à un état B ou une autre transition peut conduire à un état C selon la valeur d'une entrée *E*.

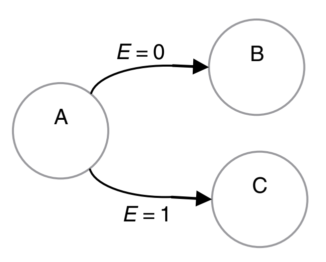


Figure 7.7 Transitions conditionnelles

Il peut y avoir maintien d'un état pour certaines valeurs d'entrée. Dans ce cas, l'arc orienté qui représente la transition se referme sur le même cercle (Fig. 7.8).

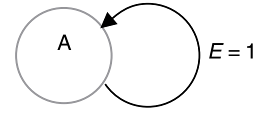


Figure 7.8 Maintien d'un état

- **Cas des machines de Moore**

  Dans une machine de Moore, les sorties évoluent après l'activation de la transition. Les valeurs des sorties sont indiquées à côté des cercles (Fig. 7.9).

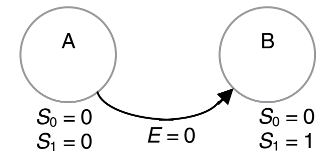


Figure 7.9 Valeurs des sorties pour une machine de Moore

- **Cas d'une machine de Mealy**

  Dans une machine de Mealy, les sorties évoluent après l'évolution des entrées. Les valeurs des sorties sont indiquées à côté des arcs orientés (Fig. 7.10).

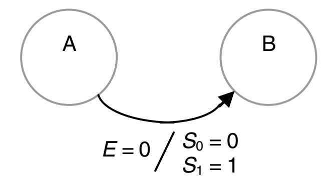


Figure 7.10 Valeurs des sorties pour une machine de Mealy

Système de serrage automatique

L'exercice porte sur un système de serrage automatique de boulons. La fonction principale FP6 permet de réaliser une mesure relative de l'angle de rotation de l'axe de serrage de la broche.
Les deux signaux *A* et *B* issus des deux capteurs angulaires placés sur l'axe de rotation de la broche de serrage contiennent deux informations : sens de rotation et évolution de l'angle de rotation.
L'information sens de rotation est contenue dans le décalage entre les deux signaux *A* et *B* (Fig. 7.11).
L'évolution de l'angle de rotation est donnée par la succession des fronts en *A* et *B* de sorte que l'apparition de chaque front corresponde à une rotation d'un degré de l'axe de rotation dans le sens du serrage ou du desserrage (Fig. 7.12).
Au sein de la fonction secondaire FS6.2, il existe deux signaux caractéristiques nommés *INC* et *DEC*. Chaque impulsion sur l'un de ces signaux correspond respectivement à une incrémentation ou une décrémentation de la mesure de l'angle de serrage.
Les chronogrammes ci-dessous (Fig. 7.13) représentent l'allure de ces signaux en fonctions des deux entrées N_A et N_B.

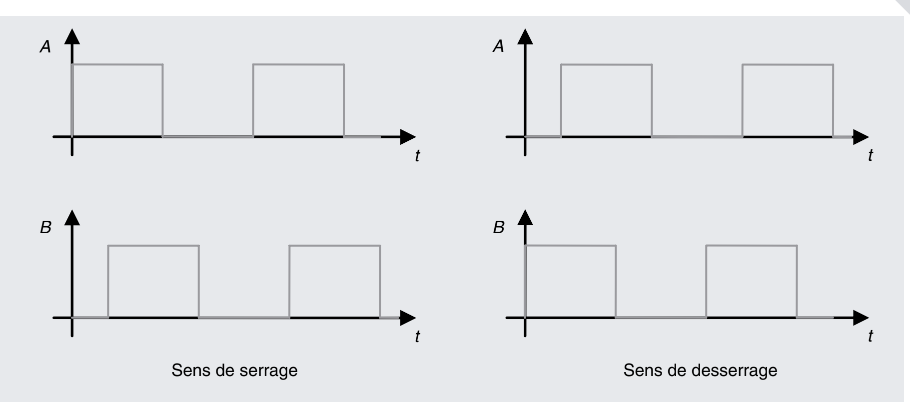


Figure 7.11 Information sens de rotation

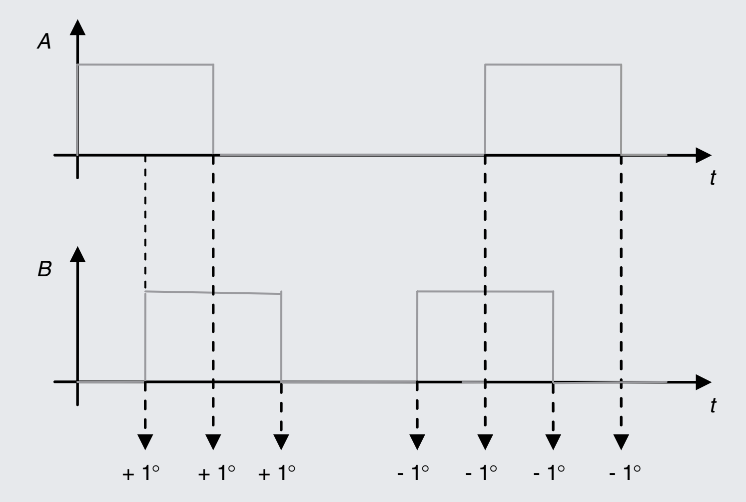


Figure 7.12 Information évolution de l'angle de rotation

Le chronogramme noté « États » représente les différents états de la machine à états qui permet de générer les signaux INC et DEC.

De façon simpliste on peut dire qu'un état correspond à une action ou un événement. Par exemple, faire évoluer une sortie correspond à une action et nécessite un changement d'état, c'est-à-dire, le passage d'un état avec sa propre action à un autre état. De même, un changement significatif d'une (ou de plusieurs des) entrée(s) correspond à un événement et implique, pour être pris en compte, un changement d'état. Le diagramme d'états ci-dessous (Fig. 7.14) est une autre représentation des chronogrammes précédents tenant compte de tous les cas possibles de changement de sens de rotation de la broche de serrage.

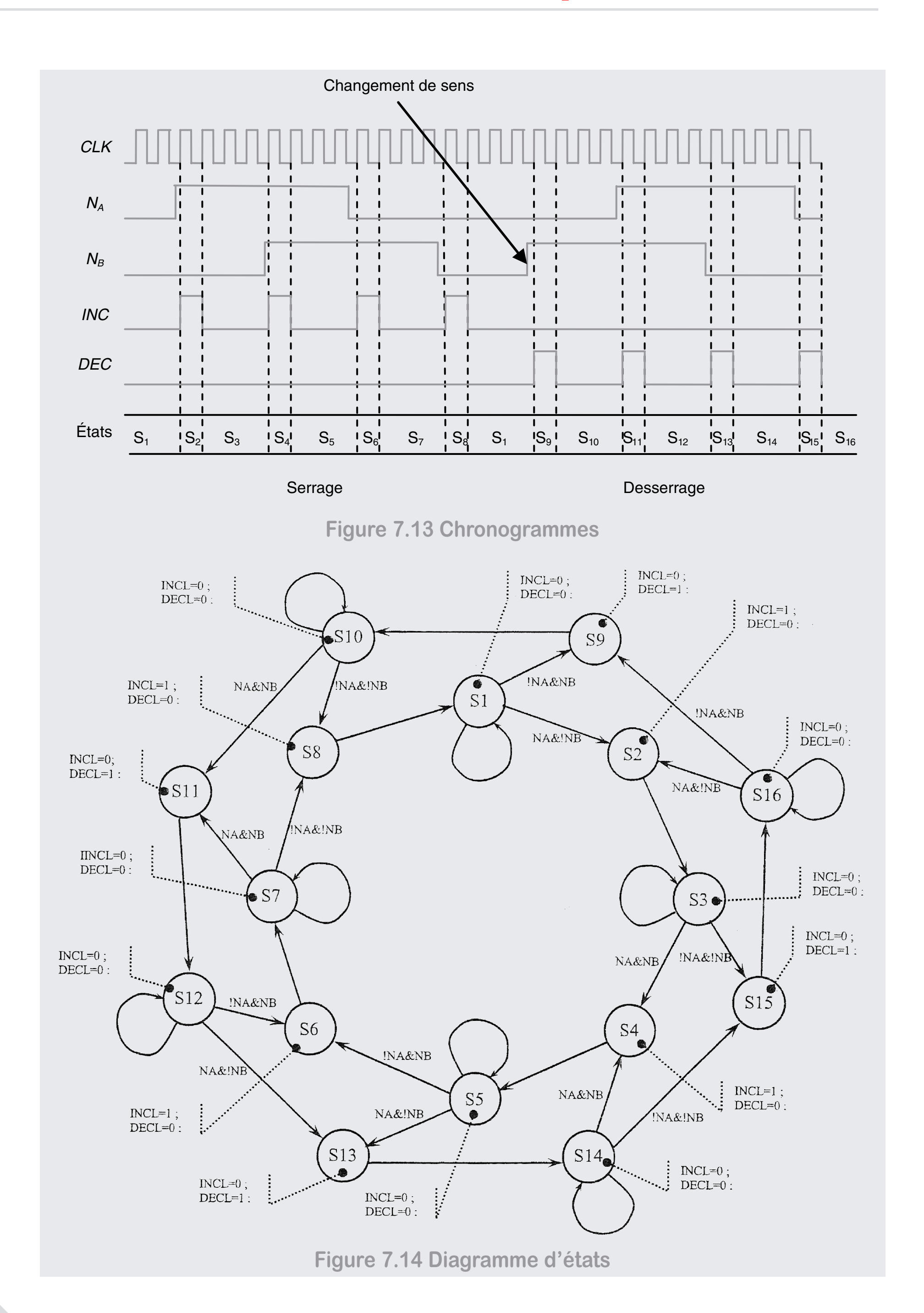


Figure 7.13 Chronogrammes

Figure 7.14 Diagramme d'états

Le diagramme donné utilise la syntaxe du langage ABEL :
– ! pour le complément,
– & pour l'opérateur logique ET,
– ; pour la fin d'une équation.

1. Le diagramme d'états comporte deux cercles d'évolution du diagramme. Quelle est la relation entre ces deux cercles et le sens de rotation de la broche ?

2. Dans ces conditions, à quoi correspondent les transitions entre les deux cercles ?

3. En fonction du nombre d'états du diagramme précédent, déterminez le nombre de bascules nécessaires à sa synthèse (transformation du diagramme d'états en un schéma structurel).

Solution

1. Le cercle du centre de S_1 à S_8 correspond au sens de serrage tandis que le cercle extérieur de S_9 à S_{16} correspond au sens de desserrage.

2. Les transitions S_1 et S_{16} entre les deux cercles correspondent aux différentes possibilités de changement de sens.

3. Un système possédant N états exige un nombre n de bascules tel que $2^n > N$. Pour 16 états, il faut donc au minimum 4 bascules.

FICHE 8

Description VHDL d'un circuit séquentiel

I Instructions du mode séquentiel

La structure de la description VHDL d'un système séquentiel est la même que pour un circuit combinatoire : l'entité puis l'architecture. La différence essentielle est que les descriptions des systèmes séquentiels utilisent des processus.

- **Définition d'un processus**

  Un processus (*process*) est une suite d'instructions exécutées séquentiellement, c'est-à-dire les unes à la suite des autres. L'exécution d'un processus est déclenchée par un changement d'état sur une variable dont le nom est défini dans la liste de sensibilité lors de la déclaration du processus. L'affectation des sorties se fait à la fin du processus.

- **Structures utilisées dans un processus**

  Un processus utilise principalement :

  – l'assignation conditionnelle :

```
if (condition) then (instruction)
else (instruction)
end if;
```

  – l'assignation sélective :

```
case (variable) is
when "00" => (instruction)
when "01" => (instruction)
when others => (instruction)
end case;
```

II Exemples de descriptions VHDL de circuits séquentiels

- **Compteur 4 bits**

Un premier exemple est la description VHDL d'un compteur 4 bits :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity COMPT_4 is
port (
    CLOCK : in std_logic;
    Q : out std_logic_vector(3 downto 0));
end COMPT_4;
architecture ARCH_COMPT_4  of COMPT_4  is
signal Q_INTERNE : std_logic_vector(3 downto 0));
begin
    process (CLOCK)
    begin
            if (CLOCK ='1' and CLOCK'event) then
            Q_INTERNE <= Q_INTERNE + 1;
            end if;
    end process;
Q <= Q_BUS_INTERNE;
end ARCH_COMPT_4;
```

Les bibliothèques définies permettent de disposer des fonctions de conversion de types qui sont nécessaires ici car l'incrémentation du compteur se fait en ajoutant 1 à la valeur décimale entière associée à la sortie *Q* qui est en binaire.
Pour incrémenter *Q*, on définit un signal interne `Q_INTERNE` car ce signal est utilisé à la fois comme entrée et comme sortie pour l'incrémentation.
Le déclenchement du processus se fait sur un changement d'état de l'horloge (CLOCK) qui est la seule variable définie dans la liste de sensibilité. L'incrémentation du signal interne se produit lors des fronts montants du signal CLOCK. À la fin, le signal interne est affecté à la sortie *Q*.

- **Bascule T**

Un second exemple est la description VHDL d'une bascule T :

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
```

```
use ieee.std_logic_unsigned.all;
entity BASCULE_T is
port (
   D,CLK : in std_logic;
   S : out std_logic);
end BASCULE_T;
architecture ARCH_BASCULE_T of BASCULE_T is
signal Q_INTERNE : std_logic;
begin
   process (CLK)
   begin
         if (CLK'event and CLK ='1') then
         if (D ='1') then
         Q_INTERNE <= not (Q_INTERNE);
         end if;
         end if;
   end process;
   Q <= Q_INTERNE;
end ARCH_BASCULE_T;
```

On retrouve les mêmes principes que dans l'exemple précédent avec, en particulier, l'utilisation d'une variable interne.

Bascule D

On considère la description VHDL d'une bascule D :

```
library ieee;
use ieee.std_logic_1164.all;
entity BASCULE_D is
port (
   D, CLK, CLR:in std_logic;
   Q, QBAR : out std_logic);
end BASCULE_D;
architecture ARCH_BASCULE_D of BASCULE_D is
signal Q_INTERNE :std_logic;
begin
   Q <= Q_INTERNE;
   QBAR <= not Q_INTERNE;
   process (CLK, CLR)
   begin
   if (CLR = '1') then
   Q_INTERNE <= '0';
   elsif (CLK'event and CLK = '1') then
   Q_INTERNE <= D;
   end if;
```

```
    end process;
end ARCH_BASCULE_D;
```

1. Quels sont les signaux d'entrée et de sortie ?
2. De quel type sont ces signaux ?
3. Quelle est la liste de sensibilité du processus ?
4. Quel est le front actif de l'horloge ?
5. Quel est le rôle de l'entrée CLR ? À quel niveau est-elle active ?
6. Donner le symbole normalisé de cette bascule.

Solution

1. Les signaux d'entrée sont D, CLK et CLR et les signaux de sortie Q et QBAR.
2. Ces signaux sont du type `std_logic` : variable logique standard qui, en dehors des valeurs 0 et 1, peut prendre un état haute impédance (Z) et un état indéterminé (–).
3. La liste de sensibilité est formée de CLR et CLK.
4. L'horloge est active sur le front montant :

```
elsif (CLK'event and CLK = '1') then
Q_INTERNE <= D;
```

5. CLR est une entrée de mise à zéro active au niveau haut :

```
if (CLR = '1') then
Q_INTERNE <= '0';
```

6. Le symbole normalisé de la bascule est donné ci-dessous (Fig. 8.1).

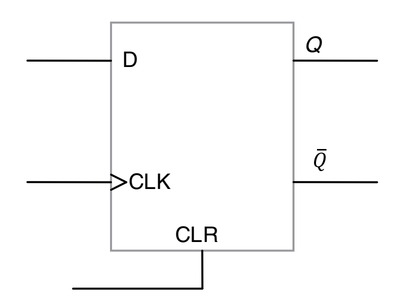


Figure 8.1 Symbole normalisé de la bascule D

FICHE 9

Paramètres électriques des circuits logiques

I Paramètres statiques

- **Tension d'alimentation**

Un circuit logique a besoin d'une alimentation continue pour lui fournir l'énergie nécessaire à son fonctionnement. La tension d'alimentation est habituellement appelée V_{CC} pour un circuit à base de transistors bipolaires et V_{DD} pour un circuit à base de transistors MOS.
Selon les cas, la tension d'alimentation autorisée pour un circuit logique est soit une valeur donnée, avec une certaine tolérance (par exemple V_{CC} = 5 V à 5 % près), soit un intervalle plus large (par exemple 3 V ≤ V_{DD} ≤ 18 V).
L'alimentation doit en général être régulée. Des condensateurs de découplages sont souvent nécessaires pour éviter les variations de tension d'alimentation lors des pics de courant qui se produisent lors des commutations. Leur nécessité, leur nombre et leur emplacement sont spécifiés dans les notices techniques des composants. Pour être efficace, un condensateur de découplage doit être implanté au plus près de la borne d'alimentation d'un circuit intégré.

- **Niveaux de tension**

La tension de sortie V_O d'un circuit logique dépend de l'état logique de la sortie, mais aussi des conditions de fonctionnement (tension d'alimentation, courant débité, etc.). Le constructeur garantit une plage de valeurs pour les tensions de sortie à l'état bas V_{OL} et à l'état haut V_{OH}, avec une tension d'alimentation V_{CC} fixée et des courants de sortie ne dépassant pas une valeur spécifiée :

– $0 \leq V_{OL} \leq V_{OL\max}$ si la sortie est à l'état bas ;

– $V_{OH\min} \leq V_{OH} \leq V_{CC}$ si la sortie est à l'état haut.

Une tension de sortie comprise entre $V_{OL\max}$ et $V_{OH\min}$ ne se rencontre donc jamais en fonctionnement normal. Si tel était le cas, cela signifierait que le circuit est défectueux ou que les intensités des courants dépassent les limites autorisées.

La tension V_I appliquée à l'entrée d'un circuit logique est interprétée comme correspondant un état bas ou un état haut selon que sa valeur se trouve dans l'une ou l'autre des plages spécifiées par le constructeur :

– $0 \leq V_{IL} \leq V_{IL\max}$ correspond à un état bas ;

– $V_{IH\min} \leq V_{IH} \leq V_{CC}$ correspond à un état haut.

Pour que les informations issues d'une sortie de circuit logique soient comprises par l'entrée d'un autre circuit logique, il faut :

$$V_{IL\max} \geq V_{OL\max} \quad \text{et} \quad V_{IH\min} \leq V_{OH\min}$$

Il faut assurer ces inégalités avec une marge de sécurité tenant de la présence éventuelle de parasites. Les écarts $V_{IL\max} - V_{OL\max}$ et $V_{OH\min} - V_{IH\min}$ sont appelés marges de bruit à l'état bas et à l'état haut (Fig. 9.1).

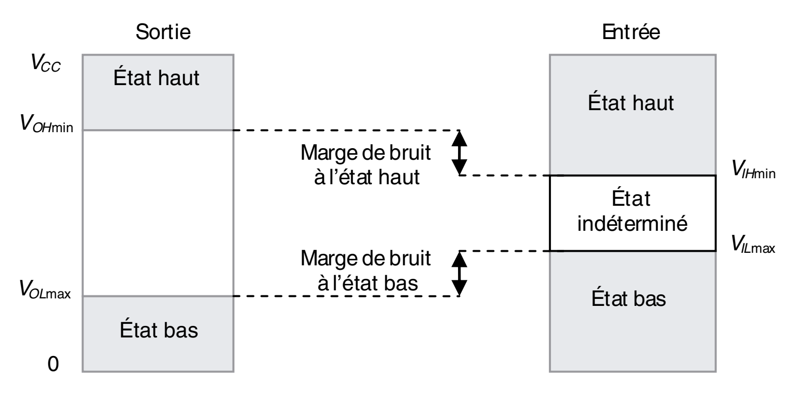


Figure 9.1 Niveaux de tension et marges de bruit

- **Courants**

Pour les différents courants, on adopte la convention de signe suivante :

– l'intensité d'un courant entrant dans une broche est positive ;

– l'intensité d'un courant sortant d'une broche est négative.

La valeur absolue de l'intensité I_O du courant dans une sortie de circuit logique ne doit pas dépasser un maximum pour que les niveaux de tension définis précédemment soient respectés. En général, cette valeur maximale n'est pas la même à l'état haut et à l'état bas. On définit donc un courant de sortie à l'état bas I_{OL} et un courant de sortie à l'état haut I_{OH}. Les courants de sortie dépendent essentiellement de la charge du circuit logique : les valeurs spécifiées ne sont que des limites à ne pas dépasser.

Une entrée de circuit logique est parcourue par un certain courant I_I dont la valeur n'est pas la même à l'état haut et à l'état bas. On définit donc un courant d'entrée à l'état bas I_{IL} et un courant d'entrée à l'état haut I_{IH}. Les courants d'entrée dépendent essentiellement du circuit logique lui-même : les valeurs spécifiées sont donc des indications sur le courant demandé par une entrée. Toutefois, ces courants ne sont pas connus avec précision : il faut donc prendre le cas le plus défavorable, c'est-à-dire celui qui correspond au courant dont l'intensité a la valeur absolue la plus grande qui puisse être observée.
Pour qu'une sortie de circuit logique puisse commander l'entrée d'un autre circuit logique, il faut que les sens des courants demandés par l'entrée correspondent au sens des courants qui peuvent être fournis par la sortie, aussi bien à l'état bas qu'à l'état haut, et que :

$$\boxed{|I_{OL}| \geqslant |I_{IL}|} \quad \text{et} \quad \boxed{|I_{OH}| \geqslant |I_{IH}|}$$

- **Sortie à collecteur ouvert ou à drain ouvert**

Une sortie à collecteur ouvert comporte un transistor bipolaire fonctionnant en commutation : l'état bas correspond au transistor saturé et l'état haut au transistor bloqué. Une résistance de rappel (appelée aussi résistance de tirage) est indispensable pour fixer la tension à l'état haut (Fig. 9.2).

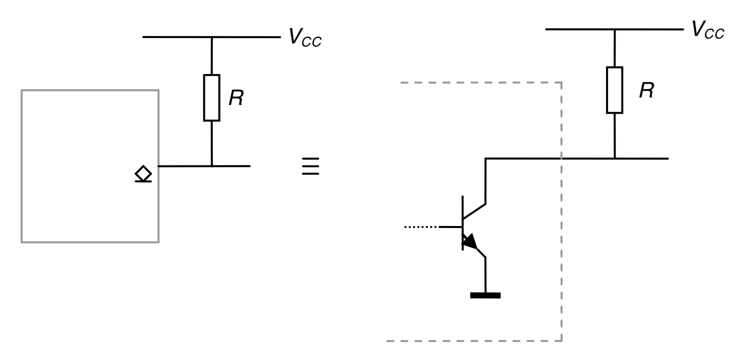


Figure 9.2 Sortie à collecteur ouvert

Quand la sortie est à l'état bas, un courant de valeur maximale I_{OL} peut traverser le transistor. Par contre, à l'état haut, le transistor est bloqué et ne peut donc fournir aucun courant. Seul un courant de fuites I_{OH} est présent.
Une sortie à drain ouvert est l'équivalent d'une sortie à collecteur ouvert, mais avec un transistor MOS.

- **Sortie trois états**

Une sortie trois états se comporte comme une sortie ordinaire quand elle est activée, mais elle possède un troisième état particulier, souvent appelé haute impédance et noté Z, où la sortie est désactivée, c'est-à-dire qu'elle se comporte pratiquement comme un circuit ouvert. Dans ce dernier, la sortie n'est parcourue que par un faible courant de fuite qui dépend du niveau appliqué par le reste du système : I_{OZL} à l'état bas et I_{OZH} à l'état haut.

II Paramètres dynamiques

- **Temps de propagation**

L'action d'un signal sur une entrée se traduit sur une sortie avec un certain retard : c'est le temps de propagation. Cette durée n'est pas forcément la même pour les fronts montants et les fronts descendants. On définit donc le temps de propagation du niveau haut au niveau bas t_{PHL} et le temps de propagation du niveau bas au niveau haut t_{PLH} (Fig. 9.3). La tension V_{ref} est une référence fixée.

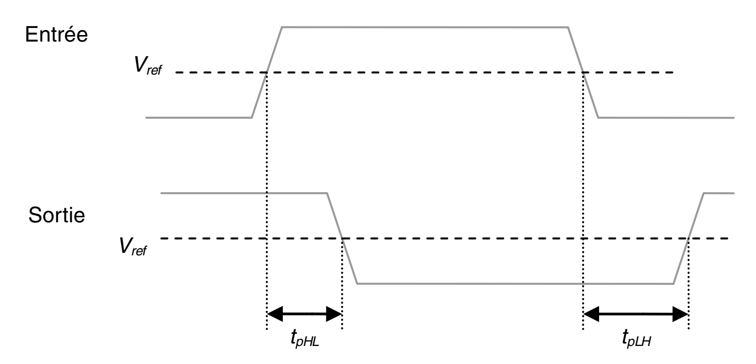


Figure 9.3 Temps de propagation

- **Temps d'activation et de désactivation (sorties trois états)**

Une sortie trois états activée comporte des temps de propagation comme une sortie ordinaire. Par contre, lors de l'activation ou de la désactivation, de nouvelles durées doivent être définies (Fig. 9.4) :

– le temps d'activation à l'état haut t_{PZH} ;

– le temps d'activation à l'état bas t_{PZL} ;

– le temps de désactivation de l'état haut t_{PHZ} ;

– le temps de désactivation de l'état bas t_{PLZ} .

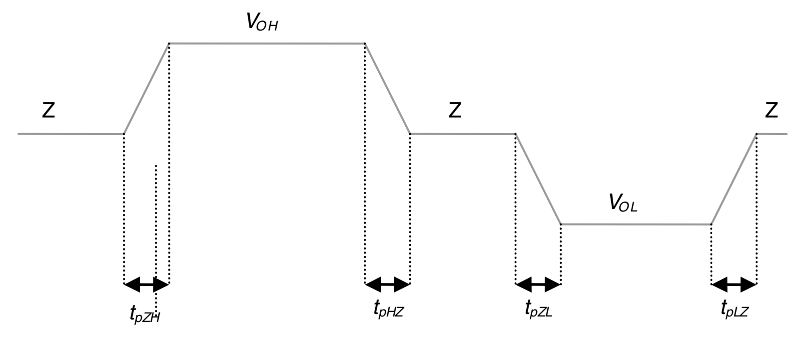


Figure 9.4 Temps d'activation et de désactivation

Remarque Par convention, on indique l'état haute impédance sur les chronogrammes par un niveau intermédiaire entre les états bas et haut. Il ne s'agit là que d'une représentation symbolique qui ne correspond pas à une valeur de tension : la sortie concernée subit le niveau de tension imposé par le reste du circuit.

- **Temps de prépositionnement et de maintien (circuits séquentiels)**

On considère un circuit possédant une entrée qui est prise en compte lors du front actif suivant de l'horloge. Le temps de prépositionnement t_{su} (*set up time*) est la durée minimale pendant laquelle la donnée présente sur l'entrée doit être stable avant le font actif du signal d'horloge. Le temps de maintien t_h (*hold time*) est la durée minimale pendant laquelle la donnée présente sur l'entrée doit rester stable après le font actif du signal d'horloge (Fig. 9.5).

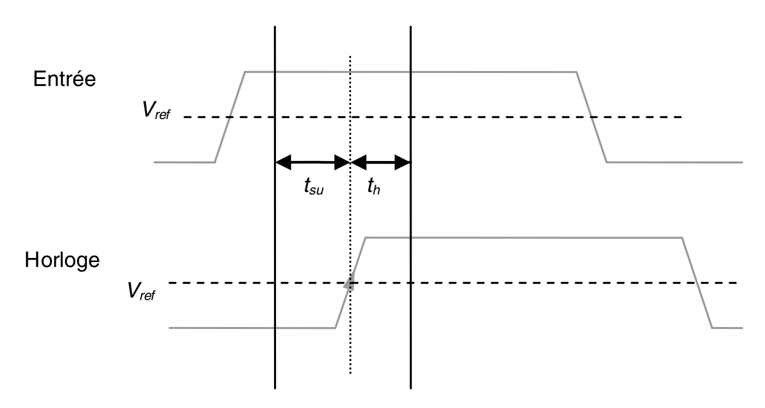


Figure 9.5 Temps de prépositionnement et de maintien

Sorties trois états

Un bus relie dix sorties trois états et quatre entrées de circuits logiques.
Les paramètres électriques des sorties trois états sont :

$$I_{OL} = 24 \text{ mA} \; ; \; I_{OH} = -2{,}6 \text{ mA} \; ; \; I_{OZL} = -20 \quad \text{A} \; ; \; I_{OZH} = 20 \quad \text{A}$$

et ceux des entrées sont :

$$I_{IL} = -0{,}4 \text{ mA} \; ; \; I_{IH} = 20 \quad \text{A}$$

1. Combien de sorties trois états peuvent-elles être activées en même temps ? Pourquoi ?
2. Quelle condition y-a-t-il sur les courants si le bus est à l'état bas ? Est-elle remplie avec les valeurs numériques données ?
3. Quelle condition y-a-t-il sur les courants si le bus est à l'état haut ? Est-elle remplie avec les valeurs numériques données ?
Conclure.

Solution

1. Une seule sortie trois états peut être activée à un instant donné, sinon, il y aurait conflit entre les différentes sorties qui voudraient imposer leur niveau de tension.
2. À l'état bas, le courant dans chacune des quatre entrées est I_{IL} (négatif donc sortant) et le courant dans chacune des neuf sorties non activées est I_{OZL} (négatif donc sortant). Tous ces courants se referment dans la sortie activée. Le courant I_{OL} de la sortie active (entrant, donc positif) doit être supérieur à la somme des valeurs absolues des courants des entrées et des sorties à l'état haute impédance :

$$I_{OL} \geqslant 4|I_{IL}| + 9|I_{OZL}|$$

On a $I_{OL} = 24$ mA et $4|I_{IL}| + 9|I_{OZL}| = 4 \times 0{,}4 + 9 \times 0{,}02 = 1{,}78$ mA. L'inégalité est donc vérifiée avec une marge importante.
3. À l'état haut, le courant dans chaque entrée est I_{IH} (positif donc entrant) et le courant dans chacune des sorties non activées est I_{OZH} (positif donc entrant). Tous ces courants se referment dans la sortie activée. Le courant I_{OH} de la sortie active (sortant, donc négatif) doit avoir une valeur absolue supérieure à la somme des courants des entrées et des sorties à l'état haute impédance :

$$|I_{OH}| \geqslant 4I_{IH} + 9I_{OZH}$$

On a $|I_{OH}| = 2{,}6$ mA et $4I_{IH} + 9I_{OZH} = 4 \times 0{,}02 + 9 \times 0{,}02 = 0{,}26$ mA. L'inégalité est donc vérifiée avec une marge importante.
Les conditions sont remplies pour que le branchement ne pose aucun problème électrique.

FICHE 10

Architecture d'un microcontrôleur

I Présentation

Un microcontrôleur est un circuit intégré rassemblant un microprocesseur et d'autres composants tels que de la mémoire et des périphériques. Il permet de réaliser des montages sans nécessiter l'ajout de composants annexes. Les avantages des microcontrôleurs sont :

– un faible encombrement,
– un circuit imprimé relativement simple,
– un coût modeste,
– une bonne fiabilité
– une bonne rapidité,
– une faible consommation.

II Description

- **Structure générale**

Un microcontrôleur est composé au minimum des éléments suivants :

– un microprocesseur (CPU pour *Central Processing Unit, unité centrale*),
– une horloge (avec éventuellement un quartz externe),
– de la mémoire,
– des interfaces parallèles pour la connexion des entrées et des sorties.

Ces différents éléments sont reliés par un ou plusieurs bus.
Deux architectures différentes existent :

– la structure de Von Neumann qui utilise une mémoire unique pour le programme et les données (Fig. 10.1) ;
– la structure de Harvard où la mémoire pour le programme est séparée de la mémoire pour les données (Fig. 10.2).

Les types exacts de mémoires dépendent du modèle de microcontrôleur. Les interfaces disponibles varient également. Outre les ports parallèles, on peut trouver :

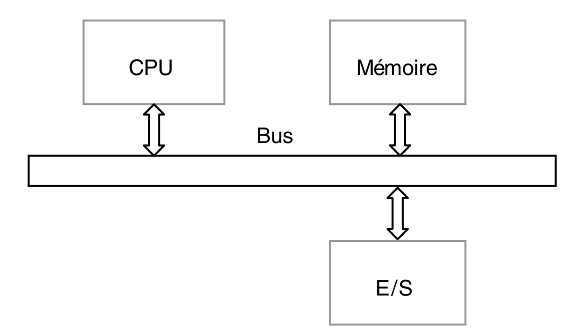


Figure 10.1 Structure de Von Neumann

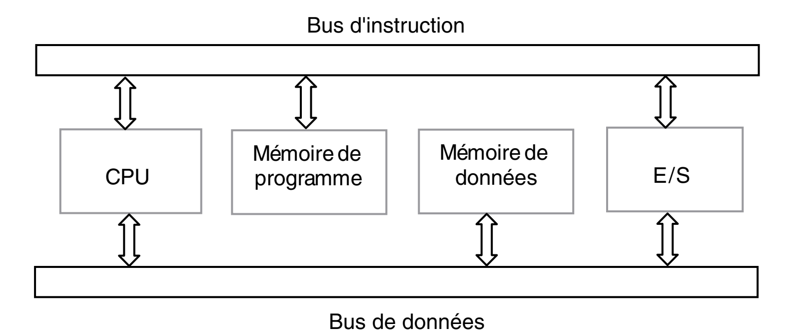


Figure 10.2 Structure de Harvard

– des interfaces série RS232 ou I2C pour la communication ;
– des compteurs (*timers*) permettent de générer des impulsions ou de mesurer des durées avec une grande précision ;
– des convertisseurs analogique-numérique pour le traitement de signaux analogiques ;
– des modules de modulation de largeur d'impulsion (PMW, *pulse-width modulation*) pour la commande de charges de puissance comme les moteurs, etc.

- **Microprocesseur**

Il exécute les instructions présentes dans la mémoire de programme de façon séquentielle. Il est constitué des éléments suivants :

– une unité arithmétique et logique (UAL ou ALU pour *Arithmetic Logic Unit*) qui effectue les opérations ;
– un séquenceur qui commande le fonctionnement du microprocesseur ;
– un décodeur d'instruction qui détermine la tâche à exécuter ;

– un compteur ordinal qui génère l'adresse de l'instruction à exécuter ou de la donnée à traiter ;
– un ou plusieurs registres accumulateurs qui contiennent temporairement les opérandes et les résultats ;
– différents registres associés aux périphériques.

Le processeur peut être à jeu d'instructions complexe (CISC pour *Complex Instruction Set Computer*) ou à jeu d'instructions réduit (RISC pour *Reduced Instruction Set Computer*).

- **Horloge**

Le signal d'horloge cadence le fonctionnement du microprocesseur. Il est produit par un oscillateur à quartz qui est souvent formé d'un circuit actif interne (inverseur et résistance) et d'un quartz et de condensateurs externes (Fig. 10.3).

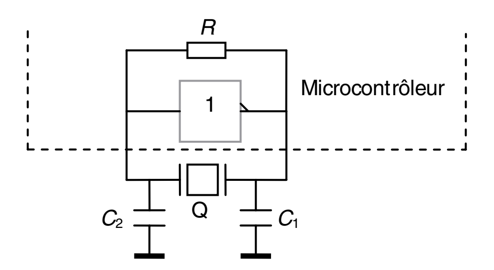


Figure 10.3 Horloge

Il existe également des horloges totalement internes. Plus la fréquence d'horloge est élevée, plus le processeur travaille vite.

- **Mémoire de programme**

Elle contient les instructions du programme. C'est une mémoire morte dont le type exact dépend du modèle de microcontrôleur (voir la fiche 11 pour des précisions sur les différentes catégories de mémoires) :
– ROM, dont le contenu est programmé par son constructeur ;
– PROM, inscrite par l'utilisateur grâce à un programmateur ;
– EPROM ou UVPROM, programmable par l'utilisateur et pouvant être effacée par un rayonnement ultraviolet ;
– EEPROM, programmable et effaçable électriquement ;
– flash, programmable et effaçable électriquement, mais plus rapide.

- **Mémoire de données**

Elle contient les données utilisées par le microprocesseur et elle est accessible en lecture et en écriture. On en trouve deux catégories (voir la fiche 11 pour des précisions sur les différents types de mémoires) :

– RAM, mémoire volatile (les données sont perdues quand l'alimentation est coupée) dont les temps de lecture et d'écriture sont courts ;
– EEPROM, mémoire non volatile (les données sont conservées quand l'alimentation est coupée) ayant un temps de lecture court, mais un temps d'écriture beaucoup plus long.

- **Interfaces parallèles**

Elles peuvent être configurées soit en entrées, soit en sorties, ce qui leur permet de :

– prendre en compte des états logiques représentant des informations externes, comme les résultats de mesure par des capteurs ;
– générer des états logiques destinés à la commande d'actionneurs.

Les interfaces parallèles sont regroupées en plusieurs ports d'entrées/sorties. La configuration et les états logiques des différentes broches sont définis dans des registres associés aux ports.

- **Interfaces série**

Une interface série permet de communiquer avec d'autres systèmes. On rencontre principalement une liaison RS232 (voir la fiche 14) ou une liaison I2C (voir la fiche 15).

- **Convertisseur analogique-numérique**

Il permet de prendre en compte des informations analogiques, par exemple des résultats issus de capteurs. Il utilise la méthode des approximations successives (voir la fiche 22).

- **Convertisseur numérique-analogique**

Il permet de produire un signal analogique (tension) à partir des états logiques internes.

- **Compteur (*timer*)**

Il sert à différentes fonctions :

– comptage d'événements,

– temporisation,
– génération d'une impulsion calibrée,
– génération d'un signal périodique.

- **Chien de garde (*watchdog*)**

Cette structure vérifie le bon déroulement du programme.

- **Bus**

On distingue trois types de bus :

– le bus d'adresses,
– le bus de données,
– le bus de commande.

III Exemple : les microcontrôleurs PIC

- **Familles de microcontrôleurs PIC**

Parmi les nombreuses références disponibles, nous prendrons comme exemple les microcontrôleurs PIC de *Microchip*. Ce sont des microcontrôleurs à architecture Harvard : les mémoires de programme et de données sont séparées. Ils utilisent des processeurs RISC, c'est-à-dire employant un jeu d'instructions réduit, ce qui accélère l'exécution du programme.
Il existe trois familles de PIC :

– *base-line* : les instructions sont codées sur 12 bits,
– *mid-range* : les instructions sont codées sur 14 bits,
– *high-end* : les instructions sont codées sur 16 bits.

- **Identification des PIC**

Un PIC est identifié par un numéro. Considérons par exemple le modèle 16F84. Les deux premiers chiffres (qui peuvent être 12, 14, 16, 17, 18) précisent la famille : 16 pour *mid-range*. Un L peut éventuellement suivre ces chiffres pour indiquer une plage de tension étendue. La lettre suivante donne le type de mémoire de programme : F signifie mémoire flash (ce serait C pour EPROM ou CR pour PROM). Les deux chiffres suivants forment un numéro d'identification. Il peut ensuite y avoir deux chiffres après un tiret indiquant la fréquence maximale du quartz (10 pour 10 MHz).

- **Description du PIC 16F84**

Le PIC 16F84 possède les caractéristiques principales suivantes :

– 35 instructions,
– 1024 mots de 14 bits de mémoire flash pour le programme,
– 68 octets de RAM,
– 64 octets d'EEPROM,
– 13 entrées-sorties (2 ports de 5 et 8 entrées-sorties),
– 1 compteur de 8 bits.

PIC 16F84A

Consulter la documentation technique du PIC 16F84A sur le site Internet de son constructeur, *Microchip* (www.microchip.com).
1. Quel est le nombre de bits de ce microcontrôleur ?
2. Quel est le type de mémoire de programme ? Comment repère-t-on cette information sur le numéro du composant ? Sur combien de bits sont codées les instructions ? Quelle est la capacité de cette mémoire (en mots puis en kilobits) ? Quel est le nombre de cycles d'effacement et d'écriture possible ?
3. Quels sont les types de mémoire de données ? Sur combien de bits sont codées les données ? Quelle est la capacité de ces mémoires (en octets puis en bits) ? Quel est le nombre de cycles d'effacement et d'écriture possible pour la mémoire morte ?
4. Quel est le nombre de bornes d'entrées-sorties ?

Solution

1. Le PIC 16F84A est un microcontrôleur 8 bits.
2. La mémoire de programme est du type flash (mémoire effaçable électriquement). C'est la signification de la lettre F dans le numéro du PIC. Les instructions sont codées sur 14 bits. La capacité de cette mémoire est de 1 024 mots, soit $1\,024 \times 14 = 14\,336$ bits. Le nombre maximal cycles d'effacement et d'écriture est de 10 000.
3. Les mémoires de données sont d'une part une RAM (mémoire vive) et d'autre part une EEPROM (mémoire programmable et effaçable électriquement). Les données sont codées sur 8 bits. La capacité de la RAM est de 68 octets, soit $68 \times 8 = 544$ bits. La capacité de l'EEPROM est de 64 octets, soit $64 \times 8 = 512$ bits. 10 000 000 cycles d'effacement et d'écriture sont possibles.
4. Le composant possède 13 bornes d'entrées-sorties réparties sur deux ports.

FICHE 11

Mémoires

I Présentation

- **Définition**

Une mémoire est un dispositif destiné à stocker des informations.

- **Mode d'accès**

On distingue :

– les mémoires à accès aléatoire où chaque mot a une adresse donnée et pour lesquelles il est possible d'accéder à une information quelconque en un temps constant ;

– les mémoires à accès séquentiel où les informations sont écrites les unes derrière les autres et pour lesquelles l'accès à un mot dépend de sa position.

II Paramètres caractéristiques

- **Capacité**

Elle représente la quantité d'informations qui peut être stockée. La capacité peut s'exprimer en bits, en octets (8 bits) ou en mots (de 16 ou 32 bits). Compte tenu des valeurs importantes des capacités, on utilise surtout les multiples de ces unités : kilobit, mégabit, kilooctet, mégaoctet... Traditionnellement, un kilobit (symbole Kb, avec un K majuscule) vaut 2^{10} = 1 024 bits et un kilooctet (symbole Ko) correspond à 2^{10} = 1 024 octets. Un mégabit (symbole Mb) vaut 2^{20} = 1 048 576 bits et un mégaoctet (symbole Mo) correspond à 2^{20} = 1 048 576 octets.

Remarque Les définitions traditionnelles ne respectent pas les normes en usage pour les autres unités. Normalement, un kilobit (symbole kb, avec un k minuscule) vaut 1 000 bits et un kilooctet (symbole ko) correspond à 1 000 octets. Pour éviter les confusions, la CEI (Commission Électrotechnique Internationale) a défini en 1998 des préfixes binaires : kibi

pour 2^{10}, mébi pour 2^{20}, etc. Ainsi un kibioctet (symbole Kio avec K majuscule) correspond à 2^{10} octets et un mébioctet (symbole Mio) correspond à 2^{20} octets. Force est de constater que ces appellations sont peu utilisées.

La capacité est liée au nombre de lignes d'adresse. Considérons par exemple une mémoire qui stocke les données sous forme d'octets et qui possède 15 lignes d'adresses. Cela correspond à 2^{15} adresses différentes, donc à 32×2^{10} octets. La capacité de cette mémoire est donc de 32 Ko, soit $32 \times 8 = 250$ Kb.

- **Temps d'accès**

  C'est la durée nécessaire à une opération de lecture ou d'écriture, c'est-à-dire le temps qui s'écoule entre l'instant où l'opération est demandée et l'instant où l'opération est terminée.

- **Temps de cycle**

  C'est la durée minimale entre deux accès successifs.

III Cycle de fonctionnement

De façon simplifiée, une mémoire pouvant être lue et écrite comporte les connexions suivantes (Fig. 11.1) :

– les entrées d'adresses ;
– les entrées et les sorties de données ;
– une entrée de choix entre lecture et écriture (R/W pour *read/write*) ;
– une entrée de sélection du circuit (CS pour *chip select*).

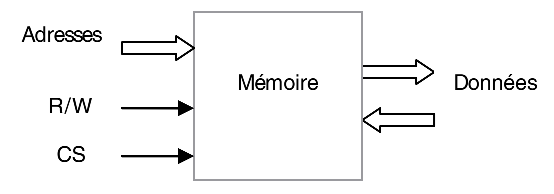


Figure 11.1 Connexions d'une mémoire

Une opération de lecture ou d'écriture s'effectue avec les étapes suivantes :

– la sélection de l'adresse ;
– le choix entre lecture et écriture (niveau appliqué sur R/W) ;
– la sélection du circuit (niveau appliqué sur CS) ;
– la lecture ou l'écriture de la donnée.

L'ensemble de ces opérations forment un cycle de lecture ou d'écriture (Fig. 11.2). Les chronogrammes font apparaître le temps d'accès t_a et le temps de cycle t_c.

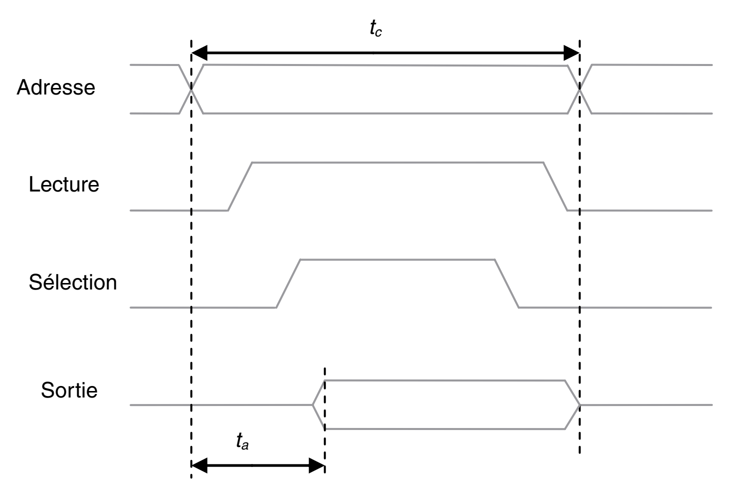


Figure 11.2 Cycle de lecture

IV Mémoires mortes

Une mémoire morte permet de conserver les informations en permanence, même quand l'alimentation est coupée.

- **Mémoire à lecture seule (ROM pour *Read Only Memory*)**

  Elle est inscrite par le constructeur au moment de sa fabrication. L'utilisateur n'a aucune action sur son contenu, il ne peut que lire cette mémoire. Les ROM ne peuvent être utilisées que pour des grandes séries puisqu'elles nécessitent une commande spéciale chez le fabricant.

- **Mémoire à lecture seule programmable par l'utilisateur (PROM pour *Programmable Read Only Memory*)**

  Les informations sont inscrites par l'utilisateur grâce à un dispositif adapté, le programmateur, qui coupe des fusibles présents dans le composant. Une PROM peut être utilisée pour des petites séries ou des prototypes, mais elle n'est programmable qu'une seule fois.

- **Mémoire à lecture seule programmable et effaçable par un rayonnement ultraviolet (EPROM pour *Erasable Programmable Read Only Memory* ou UVPROM)**

  C'est une mémoire programmable dont les informations peuvent être effacées grâce à une exposition à un rayonnement ultraviolet. Pour cela, le boîtier est muni d'une fenêtre transparente. L'EPROM doit être démontée pour être effacée. Il n'est pas possible de ne modifier qu'une partie de la mémoire. Après effacement, la mémoire peut être reprogrammée.

- **Mémoire à lecture seule programmable et effaçable électriquement (EEPROM pour *Electrical Erasable Programmable Read Only Memory*)**

  C'est une mémoire programmable dont les informations peuvent être effacées grâce à l'application d'impulsions de tension. Il n'est pas nécessaire de démonter une EEPROM pour l'effacer. La modification peut être partielle. Après effacement, la mémoire peut être reprogrammée.

- **Mémoire flash**

  C'est une mémoire effaçable électriquement dont l'écriture est plus rapide que celle des EEPROM. On ne peut cependant effacer que l'ensemble du contenu.

V Mémoires vives

Une mémoire vive (RAM pour *Random Access Memory*) permet de conserver les informations de façon temporaire, les données n'étant pas conservées quand l'alimentation est coupée. C'est une mémoire volatile à accès aléatoire. Elle est utilisée en lecture et en écriture, son contenu pouvant être modifié sans restriction.

- **RAM statique (SRAM pour *Static Random Access Memory*)**

  Les informations sont mémorisées par des bascules et elles sont conservées tant que l'alimentation est présente. Ces mémoires sont rapides, mais de capacité limitée.

- **RAM dynamique (DRAM pour *Dynamic Random Access Memory*)**

  Les informations sont mémorisées par des condensateurs. Il faut procéder à un rafraîchissement périodique à cause des courants de fuites. Ces mémoires disposent d'une plus grande capacité, mais elles sont plus lentes.

- **NOVRAM**

  C'est l'association dans un même boîtier d'une RAM et d'une EEPROM. Quand le circuit est alimenté, c'est la RAM qui fonctionne, permettant au système de bénéficier de son temps d'accès très court. Quand l'alimentation est coupée, le contenu de la RAM est transféré dans l'EEPROM. On obtient ainsi une mémoire non volatile présentant une bonne rapidité.

- **RAM ferroélectrique (FRAM pour *Ferroelectric Random Access Memory*)**

  Les matériaux ferroélectriques présentent une polarisation rémanente qui peut être inversée par l'application d'un champ électrique extérieur. On utilise cette propriété pour réaliser des mémoires RAM non volatiles. La structure ressemble à celle d'une DRAM classique, mais le diélectrique des condensateurs est une mince couche de matériau ferroélectrique.

Mémorisation des données d'une bouée de signalisation maritime

Le système considéré est une bouée de signalisation maritime. L'étude porte sur une partie de la fonction FS38 : mémoriser les données du feu. Cette fonction permet de sauvegarder les données du feu recueillies par le microcontrôleur (tension et courant (U, I) de la batterie et l'information « état du feu »), dans une mémoire à accès série par bus I2C.
On utilise un circuit FM24CL64 dont on donne un extrait de la notice (Fig. 11.3).

FM24CL64

64Kb Serial 3V FRAM Memory

Features

64K bit Ferroelectric Nonvolatile RAM
Organized as 8,192 x 8 bits
Unlimited Read/Write Cycles
45 year Data Retention
NoDelay™ Writes
Advanced High-Reliability Ferroelectric Process

Fast Two-wire Serial Interface
Up to 1 MHz maximum bus frequency
Direct hardware replacement for EEPROM
Supports legacy timing for 100 kHz & 400 kHz

A0 1 8 VDD
A1 2 7 WP
A2 3 6 SCL
VSS 4 5 SDA

Pin Names	Function
A0-A2	Device Select Address
SDA	Serial Data/address
SCL	Serial Clock
WP	Write Protect
VSS	Ground
VDD	Supply Voltage

Figure 11.3 Extrait de la notice du circuit FM24CL64

1. Préciser la technologie utilisée pour cette mémoire.
2. Donner sa capacité en Kbits, puis en octets.
3. Donner le rôle des broches : A0, A1, A2, SDA, SCL, WP.

Solution

1. Le circuit utilisé est du type FRAM (*Ferroelectric RAM*). C'est une mémoire non volatile faisant appel aux propriétés ferroélectriques d'une mince couche déposée entre deux électrodes, ce qui permet de conserver l'information grâce au phénomène de rémanence.

2. La notice du circuit indique que sa capacité est de 64 Kbits. Cela correspond à $\frac{64}{8} = 8$ Koctets, soit $8 \times 1\,024 = 8\,192$ octets.

3. La notice du circuit précise le rôle des différentes broches :

– A0, A1 et A2 : sélection du circuit ;
– SDA : passage série des données et des adresses ;
– SCL : entrée d'horloge de synchronisation ;
– WP : protection en écriture.

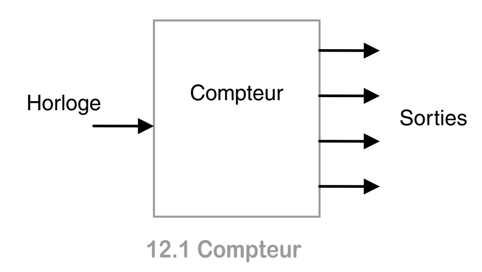
FICHE 12

Compteurs

I Présentation

- **Définition**

Un compteur est un circuit séquentiel qui permet de dénombrer des impulsions appliquées à une entrée appelée horloge. Le nombre obtenu est disponible sur les sorties dans un code donné (Fig. 12.1).



12.1 Compteur

- **Bornes disponibles**

En plus de l'entrée d'horloge et des sorties, un compteur peut comporter les bornes suivantes :

– une entrée de remise à zéro (CT = 0 ou CLR pour *CLeaR*) qui permet de placer les sorties à 0 et dont l'effet peut être synchrone (action sur un front d'horloge) ou asynchrone (action indépendante de l'horloge) ;
– une entrée de validation (EN pour *ENable*) qui autorise ou non le comptage ;
– des entrées de préchargement (DA, DB,...) qui font commencer le comptage à une valeur quelconque ;
– une entrée de choix entre le comptage et le décomptage (U/D pour *Up/Down*) ;
– une sortie de fin de comptage (CO pour *Carry Output*) indiquant la fin du cycle de comptage ;
– une sortie de fin de décomptage (BO pour *Borrow Output*) indiquant la fin du cycle de décomptage.

- **Symbole**

Le symbole d'un compteur (Fig. 12.2) fait apparaître :

– une partie inférieure comportant à gauche les sorties et éventuellement à droite des entrées de préchargement ;

– une partie supérieure comportant l'horloge, la remise à zéro et éventuellement d'autres entrées de commande.

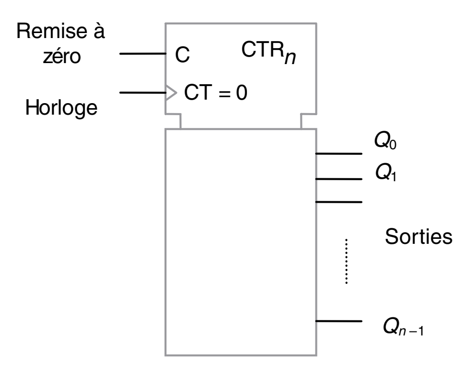


Figure 12.2 Symbole d'un compteur

- **Capacité**

C'est le nombre maximal d'impulsions qui peuvent être comptées avant un retour à l'état de départ.

- **Applications**

L'application de base est le comptage d'événements. Par exemple, il est souvent nécessaire de compter des pièces sur une chaîne de fabrication. Un capteur envoie une impulsion à chaque passage et un compteur dénombre ces impulsions (Fig. 12.3).

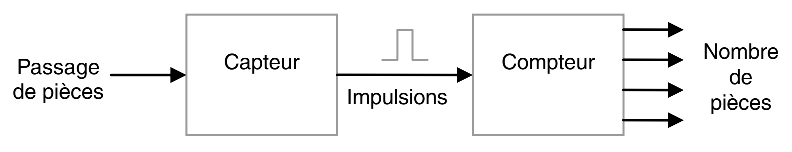


Figure 12.3 Compteur de pièces

Un compteur peut aussi être utilisé pour mesurer une durée ou pour obtenir une division de fréquence : il s'agit d'obtenir des impulsions de fréquence f/N à partir d'impulsions de fréquence f.

II Classification

Les compteurs peuvent être classés selon plusieurs critères.

- **Codage**

La sortie peut être codée de différentes façons :
– code binaire naturel ;
– code décimal ;
– code DCB, etc.

- **Sens de comptage**

On distingue :
– les compteurs pour lesquels la valeur de sortie augmente en fonction du temps ;
– les décompteurs pour lesquels la valeur de sortie diminue en fonction du temps.

- **Mode de basculement**

On distingue :
– les compteurs asynchrones ;
– les compteurs synchrones.

Les différences entre ces deux types de compteurs sont détaillées dans les paragraphes suivants.

- **Étendue des valeurs de sortie**

Il s'agit de l'ensemble des valeurs que peut prendre la sortie. L'étendue des valeurs de sortie est limitée par le nombre de bits, mais toutes les valeurs possibles pour un nombre de bits fixé ne sont pas obligatoirement utilisées.
Quand la sortie d'un compteur peut prendre N états différents, on parle de compteur modulo N.

- **Séquence de comptage**

On distingue :
– les compteurs à cycle complet qui utilisent toutes les combinaisons possibles en sortie ;
– les compteurs à cycle incomplet qui n'utilisent pas toutes les combinaisons possibles en sortie.

Un compteur modulo N avec n bits a un cycle incomplet si $N < 2^n$.

III Compteurs asynchrones

- **Définition**

Ce sont des compteurs où seul le premier étage reçoit le signal d'horloge, chaque étage étant commandé par le précédent. Ils utilisent le code binaire naturel.

- **Fonctionnement**

Prenons comme exemple un compteur binaire asynchrone trois bits à cycle complet. Il s'agit d'un compteur modulo 8.
Les différents états du compteur peuvent être indiqués dans une table de comptage (Fig. 12.4).

État	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Figure 12.4 Table de comptage d'un compteur binaire asynchrone trois bits à cycle complet

Un diagramme des états permet de représenter graphiquement le fonctionnement (Fig. 12.5).

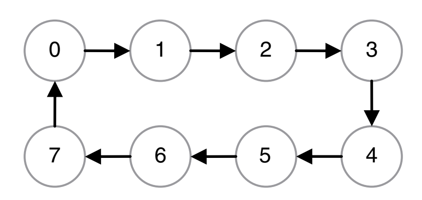


Figure 12.5 Diagramme des états d'un compteur binaire asynchrone trois bits à cycle complet

L'évolution temporelle des sorties par rapport à l'horloge est indiquée par les chronogrammes (Fig. 12.6).

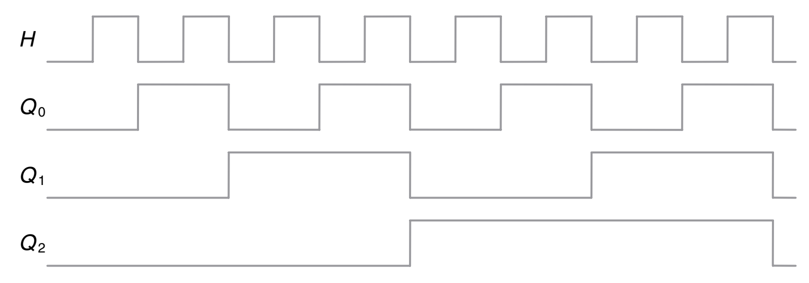


Figure 12.6 Chronogrammes d'un compteur binaire asynchrone trois bits à cycle complet

- **Propriétés**

Les compteurs asynchrones sont les plus simples, mais ils présentent plusieurs inconvénients :

– ils ne peuvent compter ou décompter qu'en binaire naturel ;
– ils sont assez lents car les durées de propagation de chaque étage s'ajoutent, ce qui conduit à une fréquence d'horloge maximale $f_{\max} = \dfrac{1}{nt_p}$ pour un compteur à étages présentant chacun une durée de propagation t_p ;
– les temps de propagation peuvent également faire apparaître des états transitoires sans signification entre deux états stables (Fig. 12.7).

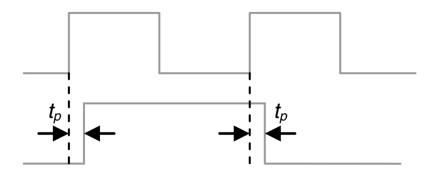


Figure 12.7 États transitoires

IV Compteurs synchrones

- **Définition**

Ce sont des compteurs dont tous les étages sont commandés par le même signal d'horloge.

- **Propriétés**

Les compteurs synchrones sont plus complexes que les compteurs asynchrones, mais ils présentent plusieurs avantages :

– ils peuvent utiliser n'importe quel code en sortie ;
– il n'y a pas de cumul des temps de propagation, ce qui conduit à une fréquence d'horloge maximale plus élevée $f_{\max} = \dfrac{1}{t_p}$;
– la durée des phases d'instabilité est limitée.

Capteur de distance à ultrasons

Un capteur de distance utilise un émetteur et un récepteur d'ultrasons placés au même point. Pour une impulsion v_e émise, le récepteur délivre une impulsion v_r d'amplitude V_{cc} = 5 V (Fig. 12.8). La durée Δt séparant les fronts montants de ces impulsions est proportionnelle à la distance d à mesurer : $\Delta t = kd$, k étant une constante.

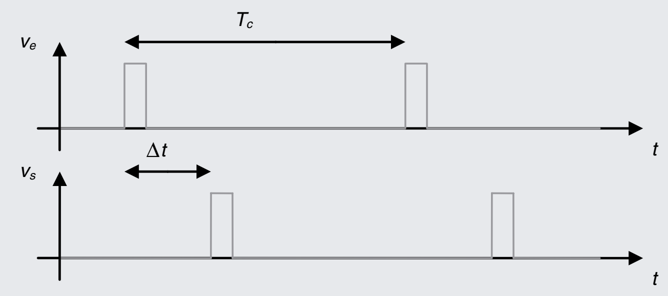


Figure 12.8 Signal émis et signal reçu

À ces impulsions de tension, il est possible de faire correspondre des signaux logiques notés V_e et V_r dont l'évolution temporelle suit l'allure décrite précédemment. La période des impulsions (d'émission et de réception) est une constante T_c. Les signaux logiques ainsi définis sont appliqués à un compteur associé à un registre parallèle 5 bits (Fig. 12.9), tous les composants logiques étant alimentés sous la même tension V_{cc}.

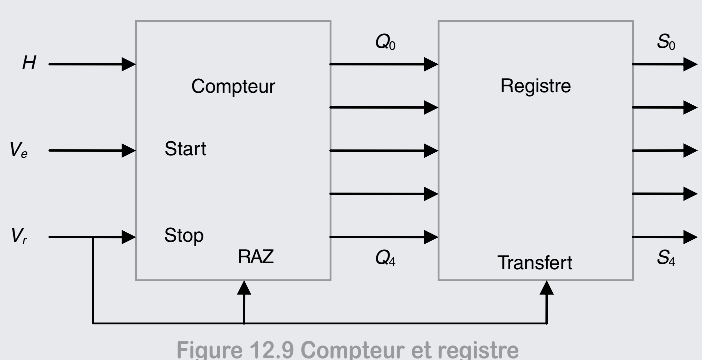


Figure 12.9 Compteur et registre

Le compteur compte avec une fréquence fixe (signal H) de période T_H. Le comptage débute lors de l'apparition d'un front montant sur la broche « *start* » et s'arrête lors de l'apparition d'un front montant sur la broche « *stop* ». Les signaux logiques de sortie, Q_0 à Q_4, représentent alors un mot numérique image de la durée de comptage, Q_0 représentant le poids le plus faible et Q_4, le poids le plus fort.
Le registre qui suit permet de retranscrire sur chaque sortie S_k la valeur de l'entrée correspondante Q_k (k variant de 0 à 4) lors de l'apparition d'un front montant sur sa broche « transfert ». Le compteur est remis à zéro lors de l'apparition d'un front descendant sur sa broche RAZ : ses sorties Q_0 à Q_4 sont alors remises à zéro.
Les durées des impulsions V_e et V_r sont très faible devant la période d'horloge T_H. Le mot numérique de sortie sera représenté sous la forme $S = [S_4\ S_3\ S_2\ S_1\ S_0]$, sa conversion en nombre entier non signé étant appelée SN.
1. Déterminer la plus petite valeur $SN_{\min}$ et la plus grande valeur $SN_{\max}$ mesurables du nombre entier SN.
2. Quelle relation doit-on imposer entre T_c et T_H afin d'obtenir une valeur numérique unique pour toute distance d et éviter ainsi toute ambiguïté dans la mesure ?
3. Déterminer l'expression de la constante k en fonction de la distance maximale $d_{\max}$ et de T_c.
Application numérique : calculer k pour $d_{\max} = 10$ cm et $T_c = 1$ ms.

Solution

1. La plus petite valeur du nombre entier SN est obtenue quand le compteur n'a pas eu le temps de compter une seule impulsion et que le mot numérique de sortie est alors [0 0 0 0 0], ce qui donne $SN_{\min} = 0$.
La plus grande valeur du nombre entier SN est obtenue quand le compteur est plein et que le mot numérique de sortie est [1 1 1 1 1], ce qui donne $SN_{\max} = 2^5 - 1 = 31$.
2. Pour obtenir une valeur numérique unique pour toute distance d, il faut assurer la condition $T_c < 2^5\, T_H$.
3. La distance maximale $d_{\max}$ correspond à une durée Δt égale à T_c. La relation de proportionnalité s'écrit dans ces conditions : $T_c = k\, d_{\max}$. On en déduit l'expression de la constante : $k = \dfrac{T_c}{d_{\max}}$.

L'application numérique conduit à : $k = \dfrac{1 \times 10^{-3}}{10 \times 10^{-2}} = 10^{-2}\ \text{s} \cdot \text{m}^{-1}$.

FICHE 13

Liaison parallèle

I Présentation

- **Principe**

Dans une liaison parallèle, N bits (8, 16, 32, voire plus) sont envoyés simultanément sur N voies différentes. Il faut $N + 1$ fils (N fils de données et un fil de masse) pour la transmission (Fig. 13.1).

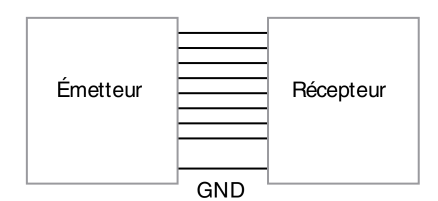


Figure 13.1 Liaison parallèle sur 8 bits

- **Propriétés**

La liaison parallèle présente l'intérêt de la simplicité. C'est aussi théoriquement la liaison la plus rapide puisque les N bits sont envoyés en même temps. Toutefois, le cadencement des processeurs avec des fréquences de plus en plus grandes a fait perdre de l'intérêt à la liaison parallèle.
La masse de cuivre nécessaire est beaucoup plus importante que pour une liaison série, ce qui augmente considérablement le coût. La transmission sur plusieurs conducteurs proches (câbles en nappe) peut être source de perturbations.
Les liaisons parallèles sont en général réservées à des transmissions à courte distance. On ne les rencontre que dans quelques domaines :

– les connexions entre composants sur les cartes électroniques ;
– les liaisons Centronics ;
– les bus d'instrumentation IEEE488 (ou GPIB) ;
– les interfaces de transmission rapide SCSI.

II Liaison Centronics

Cette interface était utilisée pour la communication entre un micro-ordinateur et une imprimante avant la généralisation des liaisons USB (Centronics est un constructeur d'imprimantes, créateur de l'interface qui porte son nom). Elle peut aussi être employée pour piloter des entrées-sorties numériques.

- **Description**

  Une liaison Centronics comporte huit fils de données (D0 à D7), un fil de masse et plusieurs fils de contrôle dont trois ont un rôle essentiel dans le processus de transmission :

  – STR (*strobe*, validation) ;
  – ACK (acknowledge, acquittement) ;
  – BUSY (occupation).

- **Protocole de communication**

  L'échange des informations se fait suivant la procédure de la poignée de main (*handshake*) qui correspond à un dialogue entre l'émetteur et le récepteur destiné à vérifier que la transmission est possible puis à confirmer son bon déroulement :

  – l'émetteur vérifie que le récepteur n'est pas occupé en regardant si le signal BUZY est à 0 ;
  – il envoie une donnée de huit bits ;
  – il prévient le récepteur qu'il a envoyé une donnée en appliquant une impulsion au niveau bas d'au moins 0,5 µs sur le fil STR ;
  – le récepteur passe alors en mode occupé en mettant le signal BUSY à 1 pendant au plus 1 ms, empêchant ainsi tout nouvel envoi de données ;
  – il envoie un signal d'acquittement (accusé de réception de la donnée) en mettant le fil ACK à 0 pendant 0,5 µs au maximum ;
  – il remet ensuite le signal BUSY à 0 pour indiquer qu'il est à nouveau disponible.

III Bus IEEE488 (GPIB)

- **Description**

  Le bus GPIB (*General Purpose Interface Bus* : bus d'interface d'usage général) a été développé en 1965 par Hewlett-Packard sous le nom de HPIB avant de se répandre comme bus d'instrumentation. La norme IEEE488 (IEEE : *Institute of Electrical and Electronic Engineers*) a ensuite précisé les contraintes mécaniques et électriques du bus en 1978 (IEEE488-1) puis certaines règles du protocole en 1987 (IEEE488-2).

Le bus IEEE488 permet de connecter entre différents dispositifs : ordinateur, instruments de mesure, etc. (quinze appareils au maximum). Il comporte un contrôleur qui supervise les échanges, un émetteur (*talker*) qui envoie des informations, et des récepteurs (*listeners*) qui reçoivent les informations. Un même appareil peut avoir plusieurs fonctions. Chaque instrument possède une adresse.

- **Lignes du bus**

Le bus comporte huit fils de données, huit fils de commande et huit fils de masse. La longueur maximale de la liaison est de 20 m et le débit maximal de transfert peut aller de 1 Mo/s à 8 Mo/s. L'échange des informations se fait suivant la procédure de la poignée de main (*handshake*).
Les huit fils de commandes se décomposent en :

• trois fils pour la procédure de *handshake* (Fig. 13.2) :
  – DAV (*DAta Valid*) : cette ligne indique quand les données sont stables et peuvent être acceptées par les instruments ;
  – NDAC (*Not DAta Accept*) : cette ligne indique si un instrument a accepté ou non un message ;
  – NRFD (*Not Ready For Data*) : cette ligne indique si un instrument est prêt ou non à recevoir un message.

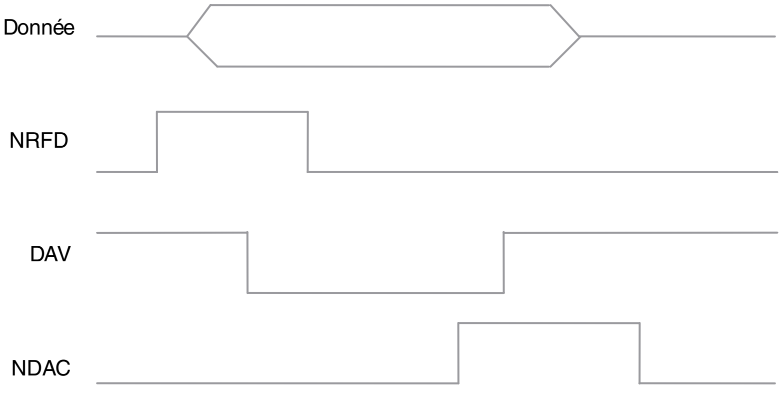


Figure 13.2 Procédure de poignée de main

• cinq fils de gestion :
  – ATN (*ATtentioN*) : quand le contrôleur active cette ligne, tous les instruments attendent une commande tandis que lorsqu'il la désactive, seuls les récepteurs reçoivent des données ;
  – IFC (*InterFace Clear*) : le contrôleur active cette ligne pour que les instruments réinitialisent leur interface GPIB ;

– REN (*Remote ENable*) : quand cette ligne est activée, les instruments sont sous le contrôle du bus tandis quand elle est désactivée, les appareils ne répondent qu'à leurs boutons de commande ;
– EOI (*End Of Identify*) : l'émetteur peut utiliser cette ligne pour signaler la fin d'un message ;
– SRQ (*Service ReQuest*) : cette ligne est activée par un instrument pour demander l'attention du contrôleur.

Protocole Centronics

1. Tracer des chronogrammes des signaux STR, BUZY et ACK lors de la transmission d'un caractère par une liaison Centronics.
2. Préciser les signaux qui sont transmis de l'émetteur vers le récepteur et ceux qui font le trajet inverse.

Solution

1. Il suffit de suivre la description du protocole donnée plus haut (Fig. 13.3).

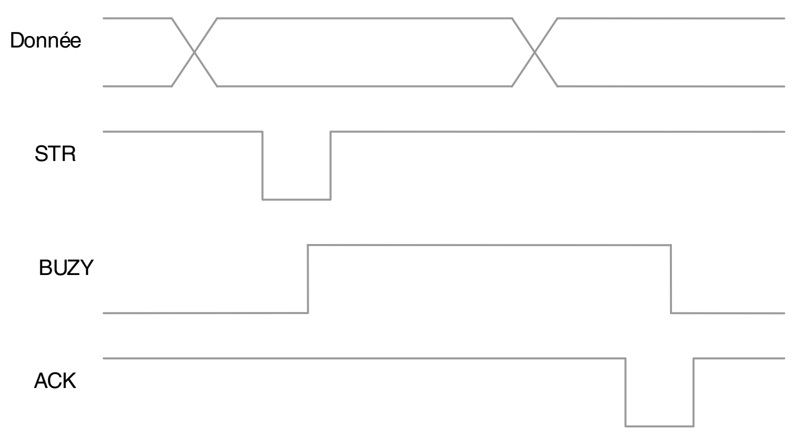


Figure 13.3 Chronogrammes du protocole Centronics

2. Les données et le signal STR sont transmis de l'émetteur vers le récepteur tandis que les signaux BUZY et ACK proviennent de récepteur.

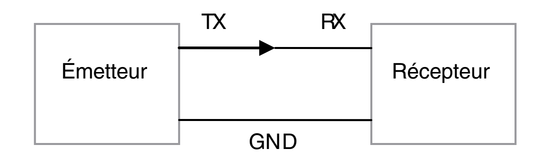
FICHE 14

Liaison série

I Présentation

- **Principe**

Dans une liaison série, les données sont envoyées successivement, bit par bit. Il suffit de deux fils (un fil de signal et un fil de masse) pour la transmission (Fig. 14.1).



TX : broche d'émission, RX : broche de réception

Figure 14.1 Liaison série

- **Liaison synchrone ou asynchrone**

Dans une liaison synchrone, l'émetteur et le récepteur sont cadencés à la même horloge. Les bits sont envoyés successivement sans séparation des caractères.
Dans une liaison asynchrone, les bits sont émis de façon irrégulière. Chaque caractère est précédé d'une information indiquant le début de sa transmission (bit de début de mot appelé souvent bit de *start*) et est suivi d'une information indiquant la fin de sa transmission (bit(s) de fin de mot appelés souvent bit(s) de *stop*). Un bit de vérification de la parité peut être présent pour détecter une éventuelle erreur de transmission (Fig. 14.2).



D : bit de début, P : bit de parité, F : bit de fin

Figure 14.2 Émission d'un caractère par une liaison asynchrone

- **Sens de liaison**

Il existe trois modes d'exploitation d'une liaison :
– simplex : les données circulent dans un seul sens, de l'émetteur vers le récepteur ;
– semi-duplex (ou *half-duplex*) : les données circulent dans un sens ou dans l'autre, mais pas en même temps ;
– duplex intégral (ou *full-duplex*) : les données circulent simultanément dans les deux sens.

II Norme RS232

- **Présentation**

La norme RS232 a été définie en 1969 par l'EIA (*Electronic Industries Alliance*). Ses caractéristiques fonctionnelles ont été reprises sous l'appellation V24 par l'UIT (Union Internationale des Télécommunications ou en anglais *International Telecommunication Union*, ITU). Elle définit les caractéristiques d'une liaison série asynchrone. La communication s'effectue point à point (un émetteur et un récepteur).

- **Niveaux électriques**

L'état logique 1 correspond à une tension inférieure à – 3 V et l'état logique 0 à une tension supérieure à 3 V. On rencontre souvent les valeurs – 12 V et 12 V.

- **Protocole**

Les caractères utilisent 7 ou 8 bits (poids faible en premier) avec ou sans bit de parité. Ils commencent par un bit de *start* à l'état logique 0 et se terminent par un ou deux bits de *stop* à l'état logique 1 (Fig. 14.3).

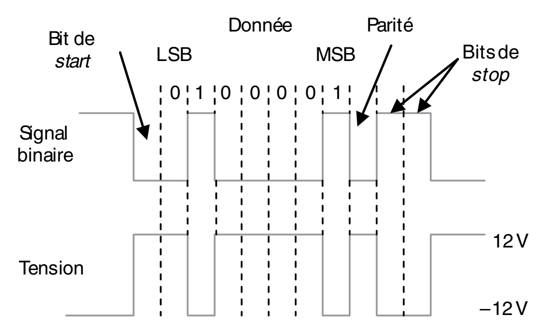


Figure 14.3 Exemple : transmission du caractère 1000010 avec bit de parité paire

Un bit de parité paire prend une valeur telle que le nombre de bits à 1 (données et parité) soit pair. Un bit de parité impaire prend une valeur telle que le nombre de bits à 1 (données et parité) soit impair.

- **Longueur maximale du câble**

La longueur maximale de la liaison dépend du débit binaire :

Débit (bps)	Longueur (m)
19 200	15,2
9600	152
4800	305
2400	915

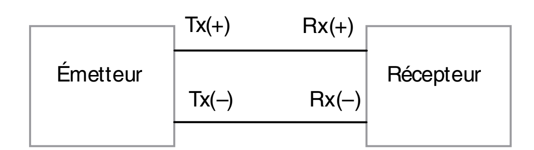
Remarque Les valeurs des longueurs maximales comportent trois chiffres significatifs, mais il ne faut pas se méprendre sur leur précision : il ne s'agit que d'estimations qui correspondent à la conversion de données d'origine anglo-saxonne, initialement exprimées en pieds (50, 100, ...).

- **Utilisation**

Les liaisons RS232 ont été largement utilisées avec les ordinateurs : ce sont les ports série aujourd'hui supplantés par les ports USB. Les liaisons RS232 restent cependant très utilisées dans l'industrie pour relier des automates, des appareils de mesure, etc.

III Norme RS485

La norme RS485 définit les caractéristiques d'une liaison série asynchrone de type différentiel qui permet un débit élevé (10 Mbps) sur une distance importante (1 200 m). Le support physique est une paire torsadée. Une liaison différentielle limite l'influence des parasites extérieurs (Fig. 14.4).



Tx(+) et Tx(–) : broches d'émission,
Rx(+) et Rx(–) : broches de réception

Figure 14.4 Liaison RS485

La communication s'effectue en mode multipoint (interconnexion de plusieurs émetteurs et récepteurs). La liaison fonctionne en semi-duplex.
La norme RS485 ne définit pas le protocole de communication.

Bouée de signalisation maritime

Le système considéré est une bouée de signalisation maritime déjà rencontrée dans la fiche 11. L'exercice porte sur la communication série RS232 entre le poste de contrôle (ordinateur) et la balise radio de télécontrôle.
Il s'agit d'analyser la fonction FS33 « Adapter les signaux de la liaison RS232 » qui permet une mise au niveau des tensions à la norme RS232 (Fig. 14.5). Les entrées sont les signaux RXA (tension bipolaire), TX (tension unipolaire) et Cmd _ICL : commande de mise en fonction du circuit. Les sorties sont les signaux RX (tension unipolaire) et TXA (tension bipolaire).

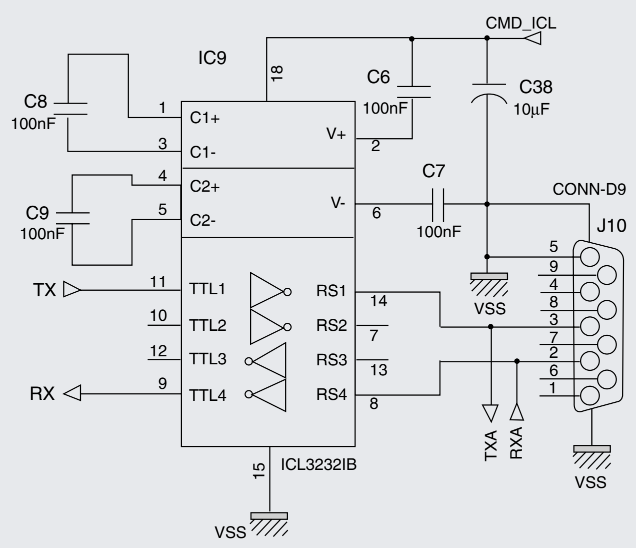


Figure 14.5 Schéma structurel de la fonction FS33

1. Donner les principales caractéristiques technologiques de cette liaison (synchrone, asynchrone, organisation d'un octet transmis).
2. Évaluer la distance maximale d'utilisation (dizaine, centaine ou millier de mètres), préciser son débit maximal.
3. La tension CMD_ICL vaut 3,3V. Consulter la documentation technique du circuit ICL3232 sur le site Internet de son fabricant, *Intersil* (www.intersil.com).
a) Préciser la tension en sortie TXA pour un niveau logique 0 en entrée TX.
b) Préciser la tension en sortie TXA pour un niveau logique 1 en entrée TX.
c) Quels sont les rôles des condensateurs C_8, C_9, C_6 et C_7 associés au ICL3232 ?



4. Analyse de relevés

La configuration de la liaison RS232 est la suivante : 1 bit de *stop*, 1 bit de parité paire, 8 bits de données.

Le signal RXA relevé sur le connecteur J10 est donné sur le chronogramme ci dessous, qui représente la transmission de deux octets (Fig. 14.6).

Rappel sur la liaison RS232 : la ligne au repos est au niveau logique 1, le bit de *start* est un passage au niveau logique 0.

a) Repérer sur le chronogramme de RXA, pour le premier octet transmis. Les intervalles de temps correspondant aux :

– bit de *start*,

– 8 bits de données (1er bit transmis D0),

– bit de parité paire,

– bit de *stop*.

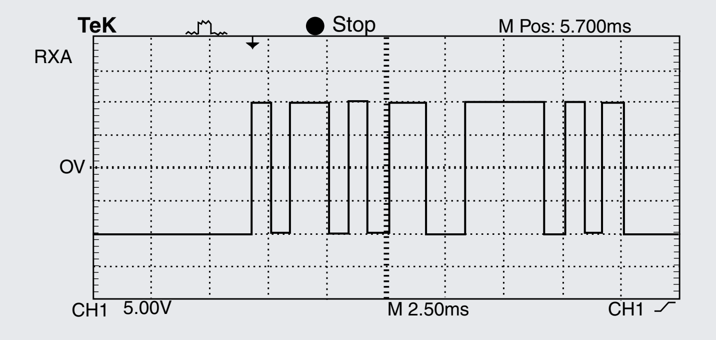


Figure 14.6 Chronogramme du signal RXA

b) Représenter le chronogramme du signal RX sur la broche 9 du circuit IC9.

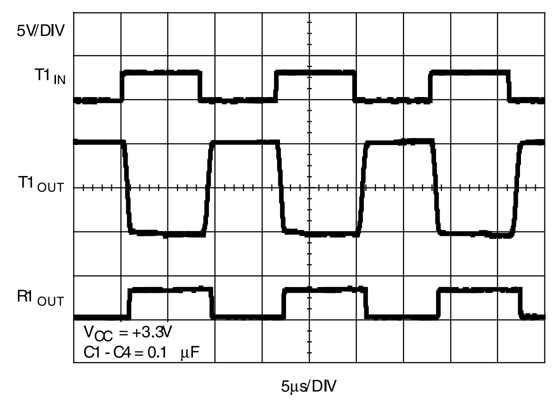
c) Préciser les valeurs en hexadécimal des deux données transmises.

Solution

1. La liaison série RS232 est une transmission série de type asynchrone. L'octet à transmettre est envoyé bit par bit (poids faible en premier). Comme il n'y a pas d'horloge commune entre l'émetteur et le récepteur, des bits supplémentaires sont indispensables au fonctionnement : bit de début de mot (*start*), bit(s) de fin de mot (*stop*). D'autre part, l'utilisation éventuelle d'un bit de parité permet la détection d'erreurs dans la transmission.

2. La distance maximale d'utilisation dépend du débit binaire. Elle est d'environ 15 m pour un débit de 19 200 bps.

3.a) La documentation du circuit ICL3232 montre que le 0 logique correspond à une tension positive dont la valeur est supérieure ou égale à + 5 V (Fig. 14.7).

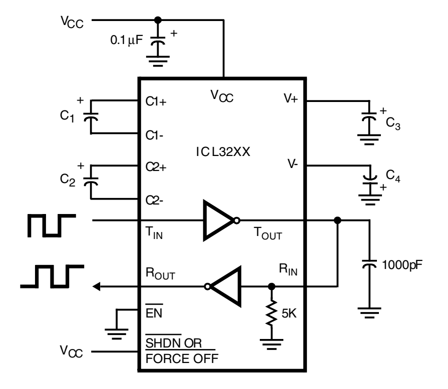


The ICL32XX maintain the RS-232 ±5V minimum transmitter output voltages even at high data rates

Figure 14.7 Extrait de la notice du circuit ICL3232

3.b) Le 1 logique correspond à une tension négative dont la valeur est inférieure ou égale à – 5 V.

3.c) Comme l'indique la notice du circuit (Fig. 14.8), les condensateurs C_6 et C_8 sont utilisés pour le doubleur de tension tandis que C_7 et C_9 font partie de l'inverseur de tension (convertisseurs à pompe de charge).



Pin Descriptions

PIN	FUNCTION
V+	Internally generated positive transmitter supply (+5.5 V).
V-	Internally generated negative transmitter supply (-5.5 V).
C1	External capacitor (voltage doubler) is connected to this lead.
C1-	External capacitor (voltage inverter) is connected to this lead.
C2+	External capacitor (voltage inverter) is connected to this lead.
C2-	External capacitor (voltage inverter) is connected to this lead.

Figure 14.8 Extrait de la notice du circuit ICL3232

4.a) Les différents bits sont repérés sur le chronogramme du signal RXA (Fig. 14.9).

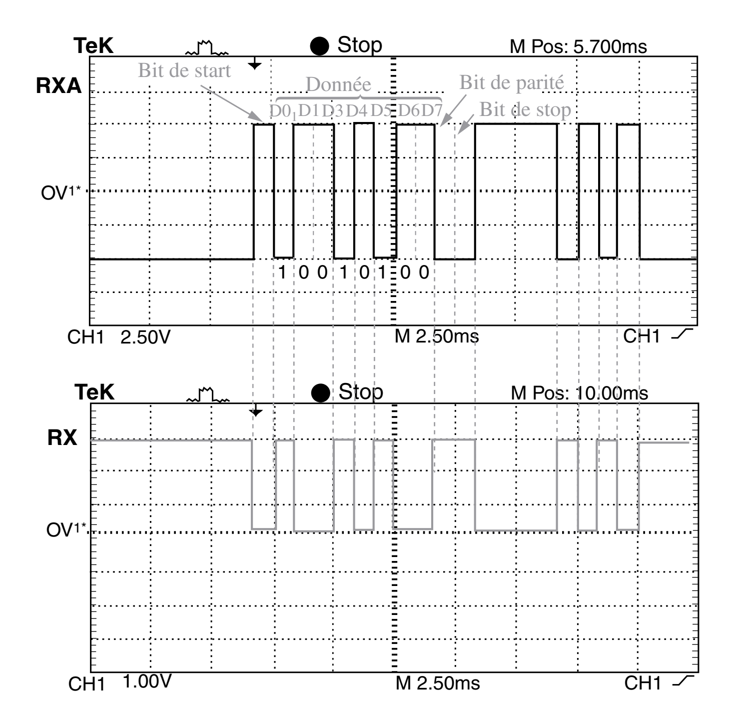


Figure 14.9 Chronogramme du signal RXA

4.b) Le chronogramme du signal RX est tracé en dessous de celui du signal RXA.
4.c) Le premier octet correspond à une valeur binaire 0010 1001, c'est-à-dire à 29 en hexadécimal. Le second octet correspond à une valeur binaire 1010 1000, c'est-à-dire à A8 en hexadécimal.

FICHE 15

Liaison I2C

I Présentation

- **Origine**

Le bus I2C (*Inter Integrated Circuit*) a été développé au début des années 1980 par Philips pour relier différents circuits électroniques de ses produits grand public. De nombreux constructeurs ont ensuite adopté ce type de liaison.

- **Caractéristiques**

Le bus I2C utilise trois fils :

– un fil pour les données (SDA) ;
– un fil pour l'horloge (SCL) ;
– un fil de masse (référence des potentiels).

Il s'agit d'une liaison série synchrone et bidirectionnelle transmettant des mots de huit bits. Le débit est de 100 kb/s en mode standard, de 400 kb/s en mode rapide et de 1 Mb/s en mode très rapide. Les applications du bus I2C concernent donc des domaines où une grande rapidité n'est pas nécessaire.

Les entrées et les sorties sont en collecteur ouvert ou en drain ouvert pour éviter les conflits électriques. Les lignes SDA et SCL doivent donc être munies de résistances de rappel qui imposent un niveau haut au repos. Le nombre de circuits reliés est limité par la charge capacitive maximale des lignes qui vaut 400 pF.

Le bus I2C assure la communication entre un maître (par exemple un microcontrôleur) et un esclave (par exemple une mémoire, un convertisseur numérique-analogique, etc.) identifié par une adresse (Fig. 15.1). Plusieurs maîtres et plusieurs esclaves peuvent être branchés sur le bus, mais le dialogue se fait entre un maître et un esclave.

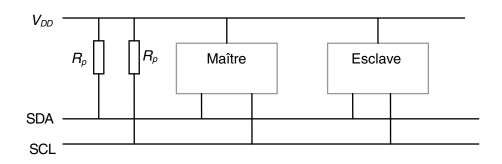


Figure 15.1 Maître et esclave



II Protocole

- **État de repos**

  Quand aucune communication n'est en cours, le bus est au repos, les circuits branchés sont à l'état haute impédance (SDA et SCL à 1) et les résistances de rappel imposent une tension égale à la tension d'alimentation

- **Prise de contrôle du bus**

  Pour pouvoir communiquer sur le bus, le circuit qui va devenir le maître doit en prendre le contrôle. C'est alors lui qui impose le signal d'horloge. Le bus doit être au repos avant la prise de contrôle : SDA et SCL à 1.

- **Condition de début et condition de fin**

  La transmission commence par une transition de l'état 1 à l'état 0 sur SDA alors que SCL est à l'état 1 (condition de début) et se termine par un passage de l'état 0 à l'état 1 sur SDA alors que SCL est à l'état 1 (condition de fin) (Fig. 15.2).



Figure 15.2 Condition de début et condition de fin

- **Transmission d'un octet**

  La transmission se fait par mots de huit bits envoyés successivement. L'émetteur commence par envoyer le bit de poids le plus fort D7 sur SDA. Il valide la donnée en appliquant un niveau 1 sur SCL. Quand SCL revient à l'état 0, l'émetteur envoie le bit suivant D6 sur SDA, et ainsi de suite jusqu'à la transmission complète de l'octet.

- **Acquittement**

  À la fin de la transmission d'un octet, l'émetteur libère la ligne SDA (c'est-à-dire que sa sortie passe à l'état haute impédance). Le récepteur doit imposer un niveau 0 pour signaler que la réception s'est déroulée correctement. C'est le bit d'acquittement (*acknowledge*).

- **Transmission d'une adresse**

Chaque esclave possède une adresse codée sur sept bits A6 à A0. La transmission de l'adresse se fait sur un octet au format suivant :

– D7 à D1 : 7 bits d'adresse ;
– D0 : bit R/W, un état 0 indique une transmission (écriture, *write*) et un état 1 indique une demande de donnée (lecture, *read*).

Dans le cas particulier d'une mémoire, il faut au moins deux octets : le premier est l'adresse du circuit et les autres correspondent à l'adresse interne.

- **Écriture d'une donnée**

La trame comporte (Fig. 15.3) :

– la condition de début,
– l'envoi de l'adresse,
– le choix du mode écriture (R/W à 0),
– le transfert des données,
– la condition de fin.

		0						
S	Adresse	R/W	A	Donnée	A	Donnée	A	P

État imposé par l'émetteur
État imposé par le récepteur

S condition de début
A bit d'acquittement
P condition de fin

Figure 15.3 Écriture d'une donnée

- **Lecture d'une donnée**

La trame comporte (Fig. 15.4) :

– la condition de début,
– l'envoi de l'adresse,
– le choix du mode lecture (R/W à 1),
– le transfert des données,
– la condition de fin.

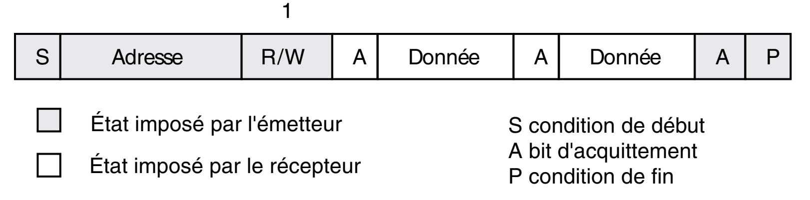


Figure 15.4 Lecture d'une donnée



Mémorisation des données d'une bouée de signalisation maritime

Le système considéré est une bouée se signalisation maritime déjà rencontrée dans les fiches 11 et 14. L'étude porte sur la deuxième partie de la fonction FS38 : mémoriser les données du feu. Cette fonction permet de sauvegarder les données du feu recueillies par le microcontrôleur (tension et courant (U, I) de la batterie et l'information « état du feu »), dans une mémoire à accès série par bus I2C (Fig. 15.5).

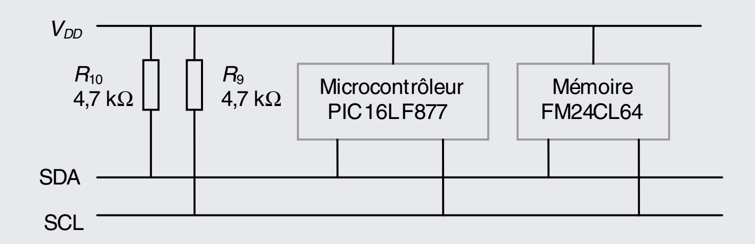


Figure 15.5 Liaison I2C entre le microcontrôleur et la mémoire

1. Donner les principales caractéristiques de ce type de bus.
2. Quel est le rôle des résistances R_9, R_{10} connectées sur les fils SCL et SDA ?
3. Justifier les valeurs de ces résistances, sachant que les caractéristiques électriques des sorties du microcontrôleur sont I_{OL} = 1mA et $V_{OL\max}$ = 0,4 V.
4. Consulter la documentation technique de la mémoire FM24CL64 sur le site Internet de son fabricant, *Ramtron* (www.ramtron.com). Repérer sur le chronogramme (Fig. 15.6) les champs : adresse MSB, données, adresse de IC6, adresse LSB.

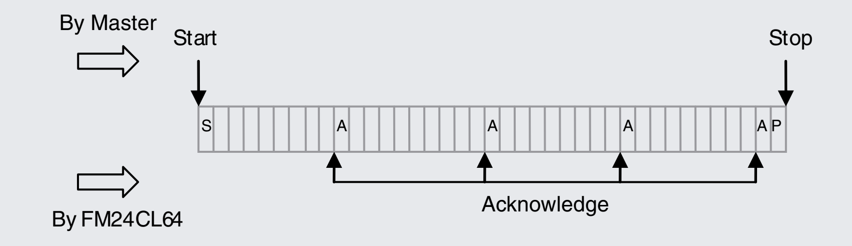


Figure 15.6 Chronogramme

Solution

1. Le bus I2C est une liaison série synchrone et bidirectionnelle qui convient à toutes les applications où la vitesse n'est pas un critère primordial.
Le bus I2C utilise trois fils :

– un fil pour les données (SDA) ;
– un fil pour l'horloge (SCL) ;
– un fil de masse.

2. R_9 et R_{10} sont des résistances de rappel indispensables car la sortie des composants I2C est à drain ouvert pour permettre le branchement de plusieurs circuits sur les mêmes fils.
3. Pour maintenir un niveau de tension correspondant à un état bas, il faut que la tension de sortie reste inférieure à $V_{OL\max}$ malgré la circulation d'un courant pouvant atteindre I_{OL}. Cela impose une valeur minimale pour les résistances R_9 et R_{10} :

$$R_{\min} = \frac{V_{DD} - V_{OL\max}}{I_{OL}} = \frac{3{,}3 - 0{,}4}{1} = 2{,}9 \text{ k}\Omega$$

La valeur choisie pour les résistances (4,7 kΩ) est bien supérieure à ce minimum, avec une marge de sécurité suffisante.
4. La documentation technique de la mémoire FM24CL64 nous fournit le chronogramme correspondant à la lecture d'un octet (Fig. 15.7).

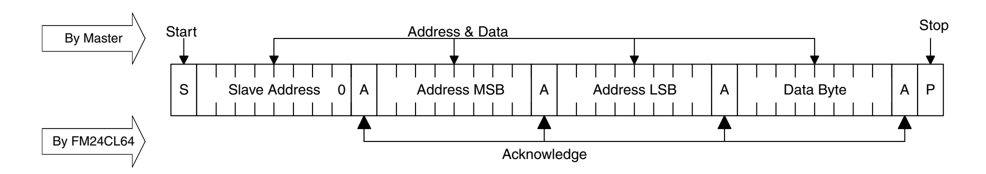


Figure 15.7 Extrait de la documentation de la mémoire FM24CL64

Les différents champs peuvent ainsi être repérés (Fig. 15.8).

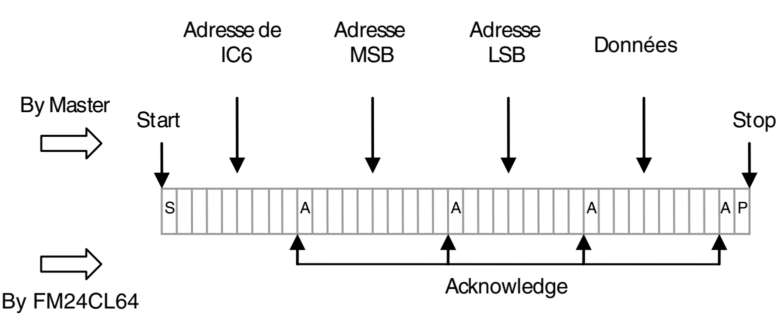


Figure 15.8 Chronogramme complété

FICHE 16

Algorithmes et algorigrammes

I Algorithme

Un algorithme est la description d'une suite finie d'actions qui permettent d'aboutir à un résultat déterminé. Il est décrit avec une notation indépendante des langages de programmation.

II Algorigramme

C'est une représentation graphique de l'algorithme qui utilise un certain nombre de symboles définis par la norme ISO 5807, *Traitement de l'information – Symboles de documentation et conventions applicables aux données, aux organigrammes de programmation et d'analyse, aux schémas des réseaux de programmes et des ressources de système*, qui date de 1985 (Fig. 16.1).

Symboles de traitement		Symboles auxiliaires	
	Symbole général Opération ou groupe d'opération sur des données, instructions, etc. pour laquelle il n'existe aucun symbole normalisé		**Renvoi** Symbole utilisé deux fois pour assurer la continuité lorsqu'une partie de ligne de liaison n'est pas représentée
	Fonction ou sous-programme Portion de programme considérée comme une simple opération		Début, fin, interruption Début, fin ou interruption d'un algorigramme
		Symboles de test	
	Entrée-sortie Mise à disposition d'une information à traiter ou enregistrement d'une information traitée	NON OUI	**Branchement** Exploitation de conditions variables impliquant le choix d'une voie parmi plusieurs

Figure 16.1 Principaux symboles utilisés dans les algorigrammes

Les différents symboles sont reliés entre eux par des lignes de liaison. Le sens général des lignes est de haut en bas et de gauche à droite. Lorsque le sens ainsi défini n'est pas respecté, des flèches indiquent le sens utilisé (Fig. 16.2).

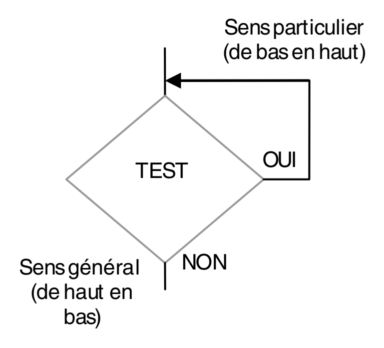


Figure 16.2 Lignes de liaison

Un algorigramme permet une visualisation commode du programme qui va être réalisé. Cependant, il n'est pas adapté à un problème complexe. La description par algorigramme devient vite lourde à manipuler et elle ne conduit pas à une bonne structuration du programme.

Mesure de période

L'exercice porte sur un dispositif de mesure de niveau de carburant dans un réservoir d'avion. Le principe en a été étudié dans l'ouvrage *Électronique analogique* de la même collection : le niveau est détecté par un capteur capacitif (fiche 2) puis l'information est transformée en fréquence (fiche 14). Il s'agit ici de mesurer la période correspondante grâce à un microcontrôleur.

Le temporisateur programmable (*timer*) du microcontrôleur est configuré en compteur 16 bits (TL0 : octet de poids faible, TH0 : octet de poids fort). Dès sa validation et tant qu'il reste validé, il compte les périodes d'horloge du microcontrôleur dont la fréquence est celle du quartz (F_{quartz} = 12 MHz) divisée par 12 (Fig. 16.3).

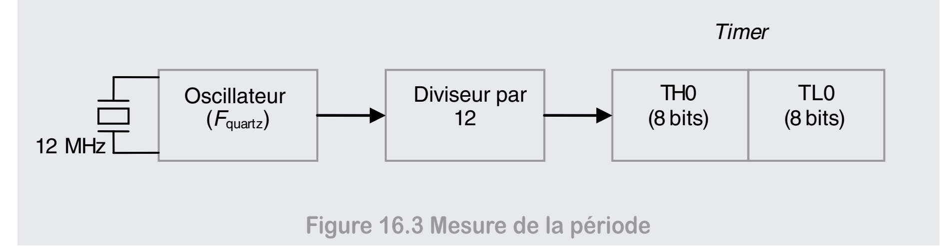


Figure 16.3 Mesure de la période

Le sous-programme de mesure, appelé dans la boucle du programme principal exécuté par le microcontrôleur est organisé de la façon suivante (Fig. 16.4).

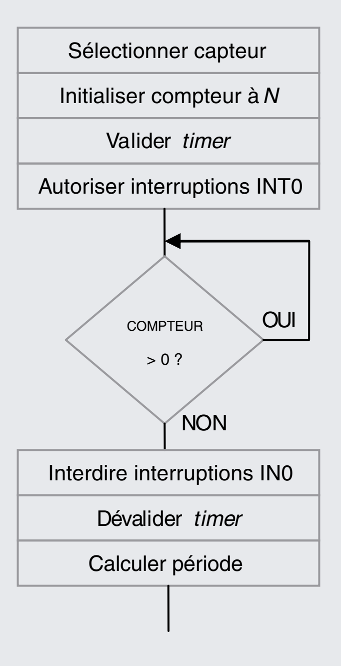


Figure 16.4 Sous-programme de mesure

COMPTEUR est une variable permettant le décompte des périodes chronométrées. *N* est le nombre de périodes à chronométrer. Les interruptions INT0 se produisent, lorsqu'elles sont autorisées, sur les fronts descendants du signal au point P7 (signal provenant du capteur sélectionné) (Fig. 16.5).

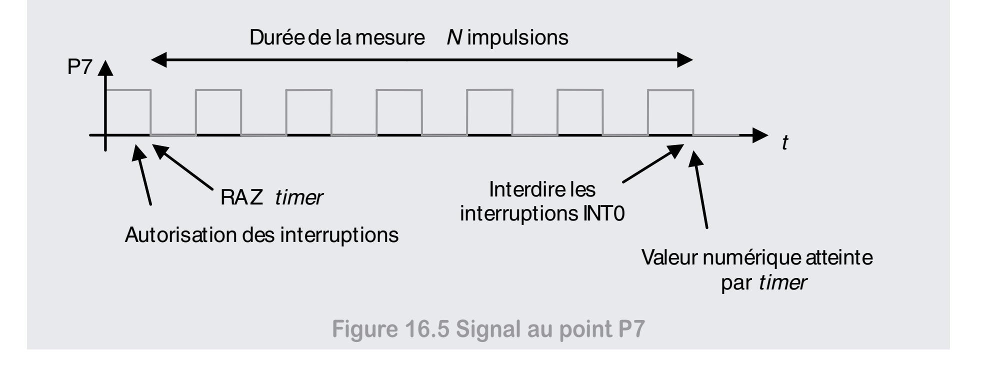


Figure 16.5 Signal au point P7

Proposer un algorigramme du programme d'interruption sachant que celui-ci doit :
– assurer la mise à zéro de TIMER lors du premier front sur INT0 (choisissez un test portant sur la valeur de COMPTEUR),
– gérer la variable COMPTEUR.

Solution

Le programme d'interruption teste la valeur du compteur en la comparant à *N* et décrémente le compteur (Fig. 16.6).

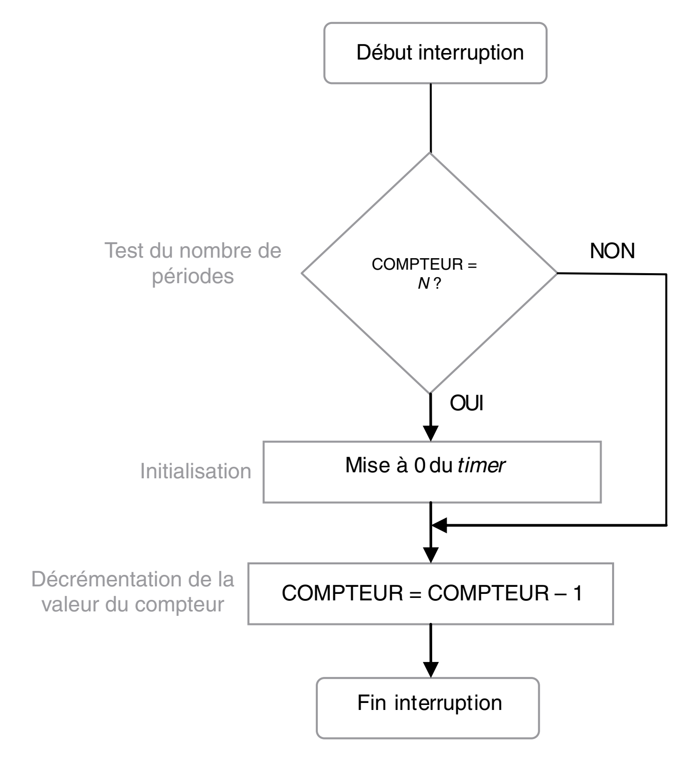


Figure 16.6 Programme d'interruption

FICHE 17

Notions d'algorithmique

I Langage de description d'algorithme

Un langage de description d'algorithme permet de décrire un algorithme en utilisant un langage proche du langage naturel et indépendant des langages de programmation. Il conduit à une définition claire et précise des opérations à effectuer et il est facilement transcrit dans un langage de programmation structurée. Le langage de description d'algorithme ne fait pas l'objet de normalisations et diffère donc selon les auteurs

II Construction d'un algorithme

- **Programmation structurée**

L'écriture d'un programme passe par les étapes suivantes :

– énoncer clairement et complètement le problème à résoudre ;
– faire intervenir des mécanismes de suite qui permettent d'aboutir au résultat ; voulu à partir des données : c'est la *formalisation* ;
– structurer ce formalisme à l'aide de schémas de programme (voir plus loin). On obtient alors l'*algorithme* ;
– traduire le mécanisme de suite à l'aide de variables. On obtient alors le programme ;
– analyser les résultats pour déceler d'éventuelles erreurs et en situer le niveau.

- **Présentation de l'algorithme**

– *Données* : ce sont les paramètres sur lesquels l'algorithme va opérer.
– *Initialisations* : c'est la définition des conditions de départ de l'algorithme. Les initialisations sont déduites de la formalisation.
– *Corps* : c'est la description des actions réalisées par l'algorithme.
– *Résultats* : c'est l'énoncé des résultats obtenus.

- **Analyse descendante**

  Le principe est d'isoler les problèmes en résolvant les plus généraux avant de descendre aux particuliers.
  On suppose *a priori* solubles les problèmes de niveau inférieur : on résoudra effectivement ces problèmes lorsque ceux du niveau supérieur seront entièrement résolus.

III Schémas de programme

Il y en a trois :

– la séquence,
– l'alternative,
– l'itération.

- **Séquence**

  C'est une suite d'opérations, ou d'instructions, à exécuter sans condition.

- **Alternative**

  Elle est de la forme :
  si < condition >
  alors < faire ceci >
  sinon < faire cela >
  fin si
  < condition > est une expression logique qui peut être vraie ou fausse.
  <faire ceci>, <faire cela> peuvent être une séquence courte ou un appel à une procédure, c'est-à-dire une demande d'exécution d'une séquence plus longue et externe.
  On peut emboîter ou mettre en cascade des alternatives :
  si < condition 1 > alors si < condition 2 > alors < faire ceci >
  sinon < faire cela >
  fin si
  sinon < faire autre chose >
  fin si
  La généralisation de l'alternative mène au schéma de choix :
  choisir selon cas
  cas 1 < faire traitement 1 >
  cas 2 < faire traitement 2 >
  cas 3 < faire traitement 3 >
  ...
  cas n < faire traitement n >
  autrement < faire traitement n+1 >
  fin cas

- **Itération**

C'est la structure de boucle, dans laquelle on répète une séquence. Une boucle doit avoir :

– une initialisation,
– au moins un critère d'arrêt réalisable.

On utilise deux types de boucles :

– la boucle à exécution inconditionnelle (itération arithmétique) qui consiste en la répétition d'une séquence un nombre prédéterminé de fois.

pour i allant de n_1 à n_2 par pas de j
faire < ceci et cela >
fin pour

i est l'indice de boucle.
n_1 est la valeur initiale de l'indice de boucle.
n_2 est la valeur finale de l'indice de boucle.
j est la valeur de l'incrément de l'indice de boucle.

– la boucle à exécution conditionnelle (itération logique) : l'entrée dans la boucle, ou la sortie de la boucle, est soumise à une condition.

Tant que < condition >	***Répéter***
Faire < plein de choses>	***Faire < tout cela >***
Fin tant que	***Jusqu'à < condition>***
La condition est testée avant l'entrée : on peut ne pas entrer dans la boucle	La condition est testée avant la sortie : on exécutera la boucle au moins une fois

Remarques Il ne faut pas sortir d'une itération par une alternative.
Il ne faut pas entrer dans une itération ailleurs qu'à son début.
Les boucles conditionnelles doivent comporter un traitement menant à la réalisation de la condition (critère d'arrêt).

Conversion en suite de caractères ASCII

Convertir un nombre binaire codé sur 16 bits, non signé, en une suite de caractères ASCII représentatifs de son équivalent en décimal. Chaque zéro non significatif sera remplacé par le caractère blanc (espace).
Exemples :
$FFFF = 65535 en décimal, soit la suite de codes ASCII :

$36	$35	$35	$33	$35

\$0210 = 528 en décimal, soit la suite de codes ASCII :

\$20	\$20	\$35	\$32	\$38

Écrire un algorithme qui résout ce problème.

Solution

Il y a trois tâches à exécuter pour parvenir au résultat :

– trouver les chiffres qui composent l'équivalent décimal du nombre à convertir,
– convertir un chiffre en son code ASCII,
– remplacer les zéros non significatifs par des espaces (caractères blancs).

1re tâche : trouver les chiffres qui composent l'équivalent décimal du nombre à convertir en caractères ASCII (conversion binaire-décimal)

Remarque
Par commodité, on écrit le nombre binaire en hexadécimal.

HEXA est le nombre binaire, codé sur 16 bits. En décimal on peut écrire :

$$0 \leqslant HEXA \leqslant 65535.$$

On pourra donc écrire *HEXA* en décimal au moyen de 5 chiffres au maximum, soit :

$$HEXA = a_4.10^4 + a_3.10^3 + a_2.10^2 + a_1.10^1 + a_0.10^0$$

On appellera, par commodité :
a_4 *DMILLE*
a_3 *MILLE*
a_2 *CENT*
a_1 *DIX*
a_0 *UNITE*
Il vient donc :
DMILLE = partie entière du quotient *HEXA*/10^4
Pour déterminer *MILLE*, il faut retrancher *DMILLE* x 10^4 à *HEXA* (on obtient le reste de la division entière précédente), puis effectuer la division entière de ce reste par 10^3. On procède de la même façon pour *CENT* et *DIX*. *UNITE* sera le reste de la dernière division entière. On obtient ainsi un premier algorithme, sous forme de séquence.

Données
    Nombre à convertir
Initialisations
    Affecter à *HEXA* la valeur du nombre à convertir
Corps de l'algorithme
Début
    Calculer la partie entière du quotient *HEXA*/10^4
    Affecter ce résultat à *DMILLE*

Calculer le reste de la division entière *HEXA*/10^4
Affecter ce résultat à *RESTE_1*
Calculer la partie entière du quotient *RESTE_1*/10^3
Affecter ce résultat à *MILLE*
Calculer le reste de la division entière *RESTE_1*/10^3
Affecter ce résultat à *RESTE_2*
Calculer la partie entière du quotient *RESTE_2*/10^2
Affecter ce résultat à *CENT*
Calculer le reste de la division entière *RESTE_2*/10^2
Affecter ce résultat à *RESTE_3*
Calculer la partie entière du quotient *RESTE_3*/10
Affecter ce résultat à *DIX*
Calculer le reste de la division entière *RESTE_3*/10
Affecter ce résultat à *UNITE*

Fin

Résultats

DMILLE = chiffre des dizaines de millier
MILLE = chiffre des milliers
CENT = chiffre des centaines
DIX = chiffre des dizaines
UNITE = chiffre des unités

On peut réécrire le corps de cet algorithme en utilisant un opérateur d'affectation (=) et en remarquant qu'on peut n'utiliser qu'une seule variable *RESTE* :

Début

DMILLE = partie entière de (*HEXA*/10^4)
RESTE = reste de la division entière (*HEXA*/10^4)
MILLE = partie entière de (*RESTE*/10^3)
RESTE = reste de la division entière (*RESTE*/10^3)
CENT = partie entière de (*RESTE*/10^2)
RESTE = reste de la division entière (*RESTE*/10^2)
DIX = partie entière de (*RESTE*/10)
UNITE = reste de la division entière (*RESTE*/10)

Fin

2e tâche : convertir un chiffre en son code ASCII

On peut appliquer deux méthodes : soit on ajoute $30 au code binaire du chiffre codé sur un octet, soit on force à 1 les bits 4 et 5 de l'octet concerné.

L'algorithme peut s'écrire :

Données

Pile contenant le code binaire des valeurs de *DMILLE*, *MILLE*, *CENT*, *DIX* et *UNITE*.

Initialisations

AdressePile à HautdePile

Corps de l'algorithme

Répéter

Forcer à 1 les bits 4 et 5 de l'octet situé à l'adresse AdressePile

Incrémenter AdressePile

Jusqu'à AdressePile=BasdePile

Résultats

Pile contenant le code ASCII des chiffres *DMILLE*, *MILLE*, *CENT*, *DIX* et *UNITE*.

3e tâche : "Effacer" les zéros non significatifs à gauche, ce qui revient à les remplacer par un espace donc de forcer à 0 le bit 5.

Il faut remplacer les zéros à gauche dans deux cas :

- ce zéro est tout à fait à gauche ;
- ce zéro est à droite d'un zéro déjà effacé, sauf si c'est celui des unités.

L'algorithme peut s'écrire :

Données

Pile contenant le code ASCII des valeurs de *DMILLE*, *MILLE*, *CENT*, *DIX* et *UNITE*.

Initialisations

Pointeur = HautdePile

Bas = BasdePile

Corps de l'algorithme

```
Si (octet à l'adresse Pointeur = $30)
Alors   Début
        Forcer à 0 le bit 5 de l'octet à l'adresse Pointeur
        Tant que (Pointeur > Bas – 2) et (octet  à l'adresse Pointeur + 1 = $30
Si (octet à l'adresse pointeur = $20)
        Alors   Début
                Forcer à 0 le bit 5 de l'octet à l'adresse Pointeur + 1
                Incrémenter Pointeur
        Fin
Fin Si
                Fin Tant que
        Fin
Fin Si
```

Résultats

Pile contenant le code ASCII des caractères à afficher.

Remarque

Tant que (Pointeur > Bas – 2) doit se comprendre Tant que (Pointeur + 1 > Bas – 1)

FICHE 18

Langage de programmation C

I Introduction

C est un langage de programmation d'usage général qui possède les outils nécessaires à une programmation structurée.
Un programme écrit en C se décompose en fonctions qui travaillent sur des variables. Il doit obligatoirement comporter au moins une fonction, la fonction *main*.
C'est un langage compilé : le programme source, écrit à l'aide d'un éditeur de texte, est traduit par le compilateur qui en fait un fichier objet exécutable par le processeur de la machine de destination.

II Compilation

Le compilateur procède en quatre étapes. La première met en forme le code source. Elle est effectuée par le préprocesseur. La deuxième étape, la compilation, traduit le code source mis en forme en code assembleur, puis vient la phase d'assemblage, la troisième étape, pendant laquelle le code assembleur est traduit en langage machine dont le code est rangé dans un fichier objet composé de plusieurs parties. La quatrième étape consiste à réunir les différentes parties obtenues dans un unique fichier au moyen de l'éditeur de liens.

III Composants de base du C

On trouve sept groupes de composants :

- les identificateurs : leur rôle est de donner un nom à une entité du programme ;
- les mots-clefs : ce sont des mots réservés pour le langage lui-même. Les spécificateurs de stockage, les spécificateurs de type, les qualificateurs de type, les instructions de contrôle constituent les principales catégories de mots-clefs ;
- les constantes : ce sont des valeurs qui apparaissent littéralement dans le code source ;
- les chaînes de caractères : ce sont des suites de caractères, tels que les chiffres et les lettres ;

– les opérateurs : ce sont des symboles qui indiquent des opérations telles que l'affectation d'une valeur à une variable, une opération arithmétique, etc. ;
– les signes de ponctuation : la virgule qui permet de séparer des entités dans une énumération ;
– les commentaires : il s'agit de texte informatif destiné aux lecteurs et usagers du programme.

IV Structure d'un programme C

Dans un programme C, on appelle expression une suite de composants de base. Suivie d'un point virgule, une expression devient une instruction. Plusieurs instructions mises entre une accolade ouvrante { et une accolade fermante } forment une instruction composée, ou bloc, équivalent à une instruction unique.
Une instruction composée d'un spécificateur de type et d'une liste d'identificateurs séparés par une virgule est une déclaration.
Un programme C se présente ainsi :

– les directives au préprocesseur ;
– les déclarations de variables externes ;
– la fonction principale, appelée *main*, composée de déclarations de variables internes et d'instructions ;
– les fonctions secondaires.

V Types prédéfinis

Le C est un langage typé : toute variable, constante ou fonction est d'un type précis, qui détermine la façon dont elle est représentée en mémoire. Les types de base en C sont les caractères, les entiers et les flottants (nombres réels).

VI Constantes

Une constante est une valeur qui apparaît littéralement dans le code source d'un programme. Il y a quatre types de constantes : entier, flottant, caractère et énumération.

VII Opérateurs

Il y a :

– l'opérateur d'affectation ;
– les opérateurs arithmétiques ;

– les opérateurs relationnels ;
– les opérateurs logiques booléens ;
– les opérateurs logiques bit à bit ;
– les opérateurs d'affectation composée ;
– les opérateurs d'incrémentation et de décrémentation ;
– l'opérateur virgule ;
– l'opérateur conditionnel ternaire ;
– l'opérateur de conversion de type ;
– l'opérateur adresse.

VIII Instructions de branchement conditionnel

Elles permettent de choisir les instructions à exécuter à partir du résultat d'un test effectué sur une expression. Il y a le branchement conditionnel, et le branchement multiple.

IX Boucles

Elles permettent de répéter une série d'instructions, autant de fois qu'il est nécessaire pour réaliser une condition. La condition peut être testée soit avant de commencer à exécuter la série d'instructions, soit après l'avoir exécutée une fois.

X Instructions de branchement non conditionnel

Il y en a deux : *break* qui permet d'interrompre le déroulement d'une boucle et de continuer l'exécution du programme à la première instruction qui suit la boucle et *continue* qui permet de passer au tour de boucle suivant, sans exécuter le reste des instructions de la boucle.

XI Fonctions d'entrées-sorties

La fonction d'écriture formatée convertit la donnée à sortir selon le format particulier choisi et la fonction de saisie formatée permet de saisir des données au clavier et de les ranger à un endroit spécifié. D'autres fonctions permettent respectivement de lire et écrire des caractères. Ces fonctions d'entrées-sorties sont dites non formatées.

XII Types composés

Ils sont construits à partir des types prédéfinis. Il y a :

- le tableau, ensemble fini d'éléments du même type ;
- la structure, suite finie d'éléments de types différents ;
- les champs de bits, c'est une structure particulière ;
- l'union, ensemble de variables de types différents ;
- l'énumération, c'est un type défini par la liste des valeurs qu'il peut prendre.

XIII Pointeur

C'est une variable qui contient l'adresse d'une autre variable. Un pointeur permet l'accès indirect à une variable.

XIV Variables

Une variable est un emplacement dans la mémoire vive (RAM) dont le contenu peut être lu et écrit. La taille de cet emplacement sera fonction du type de la variable qu'il doit contenir. On distingue les types simples dont la taille va de un octet (type char) à douze octets (type long double), les types tableaux qui regroupent plusieurs variables de type simple, les types construits, créés à partir d'un type existant et les type pointeurs qui permettent d'accéder indirectement à une autre variable.

XV Fonctions

Les fonctions sont des suites d'actions à effectuer, décrites par des instructions. Elles reçoivent des paramètres et peuvent retourner un résultat.

Mémorisation des données d'une bouée de signalisation maritime

On reprend l'étude de la liaison I2C commencée dans l'exercice de la fiche 15 et on s'intéresse ici à l'aspect logiciel.
La partie du programme à étudier permet de configurer la liaison I2C, ainsi que les modes lecture et écriture d'une donnée dans la mémoire.
On demande de :
– compléter l'espace commentaire des lignes de programme,
– compléter les chronogrammes du signal SDA répondant au programme d'écriture dans la mémoire,

– écrire une ligne de programme permettant la lecture d'un mot à une adresse donnée.

Consulter les pages 25 et 29 du document « *MPLAB C18 C Compiler Librairies* », de référence DS51297D sur le site Internet du constructeur du microcontrôleur, *Microchip* (www.microchip.com).

1. Commenter les lignes de programme ci-dessous (compléter les espaces en pointillés)

```
    OpenI2C (MASTER, SLEW_OFF);//..........................
    SSPAD = 39;
    While (1)
{
    EEByteWrite (0xA0, 0x3C, 0x00, 0xE5);//......................
}
```

2. Compléter le datagramme SDA ci-dessous (Fig. 18.1) pour l'instruction suivante :

```
EEByteWrite (0xA0, 0x3C, 0x00, 0xE5)
```

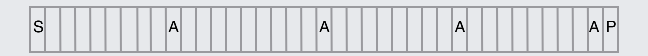


Figure 18.1 Datagramme à compléter

3. Quel est le rôle du bit A (*acknowledge*) ?

Solution

1. On utilise deux extraits du document technique (Fig. 18.2 et 18.3).
Les lignes de programme complétées sont :

```
    OpenI2C (MASTER, SLEW_OFF); //Mode maître, mode 100 kHz
    SSPAD = 39;
    While (1)
{
    EEByteWrite (0xA0, 0x3C, 0x00, 0xE5); // Écriture a l'adresse $3C00
    de la donnée $E5
}
```

2. Le datagramme est complété (Fig. 18.4) avec :
$A0 %1010 0000 (adresse de IC6)
$3C %0011 1100 (adresse MSB de la donnée)
$00 %0000 0000 (adresse LSB de la donnée)
$E5 %1110 0101 (valeur de la donnée)

3. Le bit d'acquittement (*acknowledge*) est envoyé par le composant esclave pour indiquer qu'il a bien reçu les données. La fin de réception d'un octet est signalée par la mise à 0 de ce bit.

OpenI2C
OpenI2C1
OpenI2C2

Function:	Configure the SSP*x* module.
Include:	`i2c.h`

Prototype:	`void OpenI2C( unsigned char sync_mode, unsigned char slew );`
	`void OpenI2C1( unsigned char sync_mode, unsigned char slew );`
	`void OpenI2C2( unsigned char sync_mode, unsigned char slew );`

Arguments:	***sync_mode*** One of the following values, defined in `i2c.h`: `SLAVE_7` I^2C Slave mode, 7-bit address `SLAVE_10` I^2C Slave mode, 10-bit address `MASTER` I^2C Master mode ***slew*** One of the following values, defined in `i2c.h`: `SLEW_OFF` Slew rate disabled for 100 kHz mode `SLEW_ON` Slew rate enabled for 400 kHz mode
Remarks:	`OpenI2Cx` resets the SSP*x* module to the POR state and then configures the module for Master/Slave mode and the selected slew rate.

File Name:	`i2c_open.c`
	`i2c1open.c`
	`i2c2open.c`

Code Example:	`OpenI2C(MASTER, SLEW_ON);`

Figure 18.2 Extrait du document technique de *Microchip* (p. 25)

EEByteWrite
EEByteWrite1
EEByteWrite2 (Continued)

Remarks:	This function writes a single data byte to the I^2C*x* bus. This routine can be used for any Microchip I^2C EE memory device which requires only 1 byte of address information.
Return Value:	0 if there were no errors -1 if there was a bus collision error -2 if there was a NOT ACK error -3 if there was a write collision error

File Name:	`i2c_ecbw.c`
	`i2c1ecbw.c`
	`i2c2ecbw.c`

Code Example:	`temp = EEByteWrite(0xA0, 0x30, 0xA5);`

Figure 18.3 Extrait du document technique de *Microchip* (p. 29)

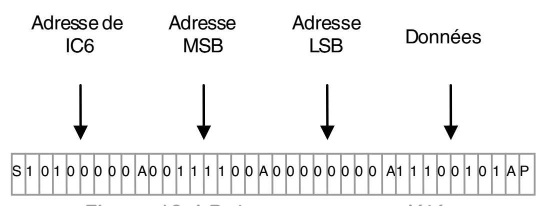


Figure 18.4 Datagramme complété

Langage assembleur

FICHE 19

I Généralités

- **Introduction**

Un assembleur est un programme qui traite les instructions d'un programme source écrit en langage d'assemblage pour en faire un programme objet en langage machine exécutable.
On utilise un cross-assembleur lorsque le programme source est écrit et édité sur une machine, l'hôte, alors que l'exécutable est destiné à une autre machine, la cible. Par exemple, l'hôte est un ordinateur PC et la cible un système construit autour d'un microcontrôleur.

- **Langage d'assemblage**

Le langage symbolique utilisé pour écrire le programme source qui sera traité par l'assembleur est appelé langage d'assemblage. C'est un ensemble de symboles mnémoniques qui représentent toutes les instructions provenant du jeu d'instructions du microprocesseur, des directives pour l'assembleur, des noms symboliques, des opérateurs et des symboles spéciaux.
Le programmeur écrit le programme source avec un éditeur de texte simple afin de produire un fichier ASCII qui pourra être traité par l'assembleur. Il doit respecter un format et une syntaxe propre au microprocesseur qui équipe le système sur lequel sera exécuté le programme objet.

- **Processus d'assemblage**

L'assembleur exécute deux passes. Durant la première, le programme source est lu afin d'établir une table des symboles. Pendant la seconde passe, le fichier objet est créé (assemblé) en tenant compte de la table des symboles. C'est durant la seconde passe qu'est produit le listing du programme source.
Chaque instruction source est traitée complètement avant la lecture de la suivante. Pendant ce processus, l'assembleur examine le champ étiquette (ou label), le champ opération et le champ opérande. Il vérifie la validité du code opération, insère le code machine correspondant dans le fichier objet et exécute les directives d'assemblage.

Toute erreur détectée par l'assembleur est signalée par un message qui précède la ligne qui la contient. Si le fichier listing n'est pas produit, un message d'erreur est affiché pour indiquer que l'assemblage ne s'est pas déroulé normalement.

II Programmes en langage d'assemblage

- **Introduction**

Les programmes écrits en langage d'assemblage sont constitués d'une suite d'instructions source. Chaque instruction source est composée d'une suite de caractères ASCII qui se termine par un retour chariot (obtenu par la touche Entrée ou Retour).

- **Format des instructions sources**

Toute instruction source peut contenir jusqu'à quatre champs : le champ étiquette (ou le caractère * dans le cas d'une ligne de commentaire), le champ opération, le champ opérande et le champ commentaire.

Champ étiquette

Il peut se présenter sous trois formes :

– une étoile (*) comme premier caractère dans le champ étiquette indique que le reste de l'instruction source est un commentaire. Les commentaires sont ignorés par l'assembleur et sont imprimés dans le listing source pour information ;

– un espace ou une tabulation en première position du champ étiquette indique que ce champ est vide. La ligne n'a pas d'étiquette et n'est pas un commentaire ;

– un symbole constitué de 1 à 15 caractères dont le premier se trouve en première position du champ étiquette constitue l'étiquette. Les caractères admis sont : les lettres minuscules et majuscules (a à z, sans accent), les chiffres 0 à 9, les trois caractères spéciaux suivants, le point (.), le dollar ($), et le souligné (_). Le premier caractère ne peut être ni le dollar ni un des chiffres. L'assembleur fait la distinction entre les majuscules et les minuscules.

Un symbole ne peut se trouver qu'une seule fois dans le champ étiquette. Dans le cas contraire, chaque référence à ce symbole sera marquée par un message d'erreur.

À l'exception de quelques directives, la valeur du compteur de programme correspondant à l'instruction ou la donnée en cours d'assemblage est affectée à l'étiquette. La valeur affectée à l'étiquette est absolue. On peut utiliser les deux

points (:) pour terminer une étiquette. Ils ne feront pas partie de l'étiquette et ne serviront qu'à séparer l'étiquette du reste de la ligne.
Une étiquette peut se trouver seule sur une ligne. Dans ce cas, l'assembleur interprète cela en affectant à l'étiquette la valeur actuelle du compteur de programme.
La table des symboles peut contenir au moins 2 000 symboles de 8 caractères. Ce nombre sera réduit si la longueur des symboles est plus grande.

Champ opération

Le champ opération vient après le champ étiquette dont il doit être séparé par au moins un caractère espace. Il doit contenir un mnémonique valable ou une directive d'assemblage. Les lettres majuscules de ce champ sont transformées en minuscules avant la vérification de la validité du mnémonique. Ainsi NOP, Nop et nop sont reconnus comme le même mnémonique.
Il y a deux types de données valides dans ce champ :

- code opération (Opcode)
  Il doit correspondre à une instruction de la machine. Le code opération comporte le nom du registre associé à l'instruction. Ces noms de registre doivent être séparés du code opération par un ou plusieurs espaces. Dans ces conditions, clra signifie mettre à 0 l'accumulateur A, mais clr a signifie mettre à 0 l'emplacement mémoire identifié par l'étiquette a.
- directive
  C'est un code opération spécial connu de l'assembleur et qui permet de contrôler le processus d'assemblage (voir directives d'assemblage).

Champ opérande

L'interprétation du champ opérande dépend du contenu du champ opération. Le champ opérande, s'il est nécessaire, doit suivre le champ opération et doit être précédé par un espace au moins. Le champ opérande peut contenir un symbole, une expression ou une combinaison de symboles et d'expressions séparés par des virgules.

Syntaxe de l'opérande pour le microcontrôleur M68HC11

Format de l'opérande	Mode d'adressage
Pas d'opérande	Inhérent ou accumulateur
<expression>	Direct, étendu ou relatif
#<expression>	Immédiat
<expression>,X	Indexé avec le registre X
<expression>,Y	Indexé avec le registre Y
<expression1> <expression2>	Mise à un ou à zéro de bit
<expression1> <expression2> <expression3>	Test de bit et branchement
><expression>	Étendu
<<expression>	Direct en page 0

Note : les parenthèses () sont utilisées pour distinguer des éléments optionnels alors que les crochets triangulaires <> indiquent qu'une expression est insérée. Ces caractères sont utilisés pour clarifier la présentation et ne font pas partie du programme source. Tous les autres caractères sont significatifs et doivent être utilisés quand c'est indiqué.

Les opérandes des instructions de manipulation de bit sont séparés par des *espaces*. <expression1> identifie l'opérande et peut indiquer soit un adressage direct, soit un adressage indexé. Dans ce dernier cas, <expression1> est suivi de ,X ou ,Y. <expression2> est le masque. C'est un octet dans lequel chaque bit mis à 1 indique qu'il est concerné par l'instruction de manipulation de bit. Il doit être codé par le programmeur. <expression3> indique le déplacement utilisé pour le branchement (adressage relatif). Le symbole > devant une expression force l'adressage étendu alors que le symbole < force l'adressage direct en page 0.

Expressions. Une expression est une combinaison de symboles, de constantes, d'opérateurs algébriques et de parenthèses. Elle détermine une valeur qui doit être utilisée comme opérande. Les expressions peuvent être constituées de symboles, de constantes ou du caractère * qui signifie « valeur actuelle du compteur de programme » reliés entre eux par un des opérateurs : + - * / % & | ^

Opérateurs. Ce sont ceux utilisés également par le langage C :

Opérateur	Signification
+	Addition
–	Soustraction
*	Multiplication
/	Division
%	Reste de la division
&	ET bit à bit
|	OU bit à bit
^	OU exclusif bit à bit

Les expressions sont évaluées de gauche à droite et il n'y a pas possibilité d'utiliser de parenthèses. Les calculs utilisent des entiers en complément à 2.

Symboles. Chaque symbole est associé à une valeur entière codée sur 16 bits utilisée à la place du symbole lors de l'évaluation de l'expression. L'astérisque * utilisé dans une expression représente la valeur du compteur de programme (adresse du premier octet d'une instruction).

Constantes. Les constantes représentent des données dont la valeur ne change pas pendant l'exécution du programme. Il y a cinq façons de représenter des constantes : sous forme décimale, hexadécimale, binaire, octale ou ASCII. Le programmeur indique le format utilisé au moyen de préfixes.

Préfixe	Format
$	Hexadécimal
%	Binaire
@	Octal
‘	ASCII

Les constantes sans préfixe sont considérées comme étant du format décimal. L'assembleur convertit les constantes en binaire et les affiche en hexadécimal dans le fichier listing.
Une constante décimale est constituée d'une chaîne de chiffres (0 à 9). Sa valeur doit se trouver dans l'intervalle 0-65535, bornes comprises.
Une constante hexadécimale comprend au maximum quatre caractères pris dans l'ensemble des chiffres (0 à 9) et le sous-ensemble (A à F) des lettres et précédés par le caractère $. Sa valeur doit se situer dans l'intervalle $0000- $FFFF, bornes comprises.
Une constante binaire est constituée d'au maximum seize caractères pris dans le sous-ensemble des chiffres (0 1) précédés du signe %.
Une constante octale comprend au plus six chiffres, pris parmi 0, 1, 2, 3, 4, 5, 6 et 7) et précédés de @. Sa valeur doit se situer dans l'intervalle @0-@177777, bornes comprises.
Un caractère ASCII peut être utilisé comme constante dans les expressions, s'il est précédé par une apostrophe. Si plusieurs caractères suivent l'apostrophe, seul le premier est pris en compte et codé par l'assembleur. Il n'y a pas de message d'erreur.

Champ commentaire

C'est le dernier champ de l'instruction source. Il est optionnel et doit être séparé du champ opérande, ou du champ opération s'il n'y a pas d'opérande, par au moins un espace. Il peut contenir tout caractère ASCII imprimable. Il est ignoré de l'assembleur et apparaît dans le fichier listing dans un but de documentation.

- **Fichiers produits par l'assembleur**

Il y en a deux : le fichier listing et le fichier objet. Le fichier listing, qui est un fichier texte, peut être composé du listing du fichier source assemblé, de la table des symboles et de la table des références croisées. Le fichier objet est au format Motorola S-Record (S1 S9).
Une ligne du fichier listing du source assemblé se présente comme ceci :

Numéro de ligne Adresse Octets du code objet [Nombre de cycles] Ligne source

Numéro de ligne est un nombre décimal de 4 chiffres, utilisé dans les références croisées.
Adresse est la valeur en hexadécimal de l'adresse du premier octet du code objet de l'instruction.
Octets du code objet est le résultat, en hexadécimal, de l'assemblage du code opération et de l'opérande associé. S'il y a plus que six octets, les octets supplémentaires sont listés sur les lignes suivantes, sans adresse.
Nombre de cycles est optionnel. Il donne le nombre de cycles nécessaire pour exécuter l'instruction et n'est présent que si l'option c a été activée.

Ligne source est la copie de la ligne du programme source.
La table des symboles sera sortie si l'option s a été activée. Elle se présente comme une succession de lignes comportant :
Symbole Adresse
Symbole provient du champ étiquette du programme source. **Adresse** est l'adresse en hexadécimal de l'emplacement référencé par le symbole.
La table de références croisées sera sortie si l'option *cre* a été activée. Elle se présente comme une succession de lignes comportant :

Symbole Adresse Emplacement1 Emplacement2 Emplacement3 ...

Symbole et **Adresse** ont la même définition que celle donnée pour la table des symboles. **Emplacement1** est le numéro de la ligne où le symbole a été défini, **Emplacement2 Emplacement3** et les suivants indiquent les numéros des lignes où le symbole est utilisé.

III Directives d'assemblage

- **Introduction**

Les directives d'assemblage sont des instructions données au programme assembleur, plutôt que des instructions à traduire en code objet. Ce paragraphe décrit les différentes directives qui peuvent être utilisées par l'assembleur. On utilisera les notations suivantes :

– ( ) : les parenthèses signalent un élément optionnel

– XYZ : les noms des directives sont en lettres majuscules

– < > : les noms des éléments sont en lettres minuscules. Ils sont contenus dans les crochets triangulaires. Tout ce qui n'est pas entre crochets doit être spécifié tel quel. Par exemple, l'élément syntaxique (<nombre>,) exige une virgule si l'élément optionnel <nombre> est choisi. On utilisera les éléments suivants par la suite :

<commentaire>	Champ de commentaire
<étiquette>	Étiquette d'une instruction
<expression>	Expression pour l'assembleur
<nombre>	Constante numérique
<chaîne>	Chaîne de caractères ASCII
<délimiteur>	Délimiteur de chaîne
<option>	Option de l'assembleur
<symbole>	Symbole de l'assembleur

- **BSZ (Block Storage of Zero)**

(<étiquette>) BSZ <expression> (<commentaire>)

La directive BSZ (ou ZMB) impose à l'assembleur de réserver un bloc d'octets et d'assigner à chacun de ces octets la valeur 0. Le nombre d'octets est donné par <expression> qui ne doit ni contenir de symbole défini plus loin dans le programme ou de symbole non défini, ni être nul sous peine de provoquer une erreur à l'assemblage.

- **EQU (EQUate symbol to a value)**

(<étiquette>) EQU <expression> (<commentaire>)

La directive EQU assigne la valeur de <expression> à l'étiquette. On ne peut pas redéfinir l'étiquette ailleurs dans le programme. <expression> ne peut contenir de symbole défini plus loin dans le programme (*Phasing error*) ou de symbole non défini.

- **FCB (Form Constant Byte)**

(<étiquette>) FCB <expression> (,<expression>,....,<expression>) (<commentaire>)

La directive FCB peut avoir un opérande ou plusieurs qui sont alors séparés par des virgules. La valeur de chaque opérande est tronquée à 8 bits puis est rangée dans un octet du programme objet. L'opérande peut être une constante numérique, une constante caractère, un symbole ou une expression.

- **FCC (Form Constant Character string)**

(<étiquette>) FCC <délimiteur> <chaîne> <délimiteur> (<commentaire>)

On utilise la directive FCC pour stocker en mémoire une chaîne de caractères. Le premier octet est stocké à l'adresse courante; L'étiquette se voit assigner la valeur de l'adresse du premier octet de la chaîne. Tout caractère imprimable peut se trouver dans la chaîne qui est enfermée entre deux délimiteurs identiques. Le délimiteur est le premier caractère ASCII imprimable autre que l'espace situé après la directive FCC.

- **FDB (Form Double Byte constant)**

(<étiquette>) FDB <expression> (,<expression>,....,<expression>) (<commentaire>

La directive FDB peut avoir un opérande ou plusieurs qui sont alors séparés par des virgules. La valeur codée sur 16 bits de chaque opérande est rangée dans deux

octets consécutifs du programme objet. Le rangement commence à l'adresse courante et l'assembleur assigne à l'étiquette la valeur de l'adresse courante.

- **FILL (FILL memory)**

(<étiquette>) FILL <expression>,<expression> (<commentaire>

La directive FILL indique à l'assembleur d'initialiser une zone mémoire avec une valeur constante. La première expression donne la valeur de la constante et la deuxième expression donne le nombre d'octets successifs à initialiser. La première expression doit se trouver dans la gamme de valeurs 0-255. Les deux expressions ne peuvent comporter ni de symbole défini plus loin dans le programme ni de symbole non défini.

- **OPT (assembler output OPTions)**

(<étiquette>) OPT <option>(,<option>,...,<option>) (<commentaire>)

La directive OPT sert à fixer le format du fichier de sortie. Les options définies par cette directive sont prioritaires sur celles définies sur la ligne de commande qui a lancé l'assembleur. Les options doivent être écrites en caractères minuscules.

c Autorise l'affichage du nombre de cycles dans le listing. Le nombre total de cycles nécessaire à l'exécution de l'instruction sera affiché après le dernier octet de l'instruction, entre crochets rectangulaires.

cre Autorise l'édition d'une table de référence croisée à la fin du fichier listing. Cette option doit être spécifiée avant le premier symbole dans le fichier source.

l Édite le listing à partir de ce point. Le format du listing est donné en annexe.

Noc Interdit l'affichage du nombre de cycles nécessaire à l'exécution de l'instruction. C'est une valeur par défaut.

Nol N'édite plus de listing à partir de ce point. C'est une valeur par défaut.

S Édite une table des symboles à la fin du fichier listing.

- **ORG (set program counter to ORiGin)**

(<étiquette>) ORG <expression> (<commentaire>)

La directive ORG remplace le contenu du compteur de programme par celle donnée par <expression>. Les instructions suivantes seront assemblées dans les emplacements mémoire situés à partir de la nouvelle adresse contenue dans le compteur de programme. S'il n'y a pas de directive ORG dans un programme source, le compteur de programme est initialisé à la valeur 0. <expression> ne peut contenir ni de symbole défini plus loin dans le programme ni de symbole non défini.

- **PAGE (top of PAGE)**

(<étiquette>) PAGE

Cette directive provoque un saut de page dans le fichier listing.

- **RMB (Reserve Memory Bytes)**

(<étiquette>) RMB <expression> (<commentaire>)

La directive RMB réserve un nombre d'octets égal à la valeur de <expression>. Ces octets ne sont pas initialisés à une valeur donnée. <expression> ne peut contenir ni de symbole défini plus loin dans le programme ni de symbole non défini. On utilise cette directive pour réserver une zone de « brouillon ».

- **ZMB (Zero Memory Bytes)**

(<étiquette>) ZMB <expression> (<commentaire>

La directive ZMB (ou BSZ) impose à l'assembleur de réserver un bloc d'octets et d'assigner à chacun de ces octets la valeur 0. Le nombre d'octets est donné par <expression> qui ne doit ni contenir de symbole défini plus loin dans le programme ou de symbole non défini, ni être nul sous peine de provoquer une erreur à l'assemblage.

FICHE 20

Développement de microcontrôleur avec MPLAB

I Présentation

Un environnement de développement est nécessaire pour développer une application utilisant un microcontrôleur. Dans le cas des PIC, ce rôle est joué par l'environnement de développement MPLAB de *Microchip*, disponible gratuitement sur le site Internet du constructeur (www.microchip.com). Avec les outils qu'il intègre on peut :

– écrire les fichiers contenant le code source ;
– compiler, assembler et lier les fichiers sources ;
– déboguer le code exécutable à l'aide d'un simulateur et d'un émulateur temps réel ;
– faire des mesures temporelles avec le simulateur ou l'émulateur ;
– visualiser les variables ;
– programmer les PIC.

Le logiciel permet la saisie du programme puis sa vérification. Le simulateur autorise l'affichage de l'état des registres et des variables en temps réel, l'exécution en mode pas à pas, la mise en place de points d'arrêts, etc. Le langage de base de la programmation est l'assembleur, mais il possible d'ajouter des modules afin de pouvoir programmer en langage C.

II Procédure

- **Créer un projet**

  Un projet contient, sous forme de fichiers, toutes les informations nécessaires pour construire une application : choix du processeur, outils logiciels utilisés, nom du projet et fichier source.

- **Écrire et sauvegarder les fichiers source**

  À l'aide de l'éditeur intégré il faut saisir le texte des fichiers en respectant la syntaxe de l'assembleur puis le sauvegarder dans un fichier dont l'extension est .asm. Ce fichier est alors ajouté au projet.

- **Construire le projet**

  La construction du projet provoque l'appel de la procédure d'assemblage. En cas de problème dû à une erreur de syntaxe, la position de la ligne de programme en erreur est signalée. Quand l'assemblage est terminé le fichier *debug* est chargé. Ce fichier sera utilisé pour simuler le comportement du programme.

- **Simuler le comportement du programme**

  Il faut configurer le simulateur puis exécuter le programme. Une fenêtre d'observation montre l'évolution des variables et du contenu des registres utilisés dans le programme. Il est possible d'insérer des points d'arrêt de façon à visualiser le contenu des registres et les variables à un moment donné de l'exécution du programme. Un mode particulier, le mode *Trace*, permet d'enregistrer les actions du programme pour visualiser leur effet ensuite. On peut créer des stimuli qui simulent l'influence de l'environnement sur le programme.

- **Programmer un composant**

  MPLAB permet la programmation de composants *in situ*. Le module ICD2 est nécessaire.

FICHE 21

Numérisation des signaux

I Chaîne de traitement numérique

Une chaîne de traitement numérique comporte autour d'un calculateur (Fig. 21.1) :

– un dispositif d'acquisition comprenant un échantillonneur-bloqueur, souvent précédé d'un filtre anti-repliement, et un convertisseur analogique-numérique ;
– un dispositif de restitution comprenant un convertisseur numérique-analogique et un filtre de lissage.

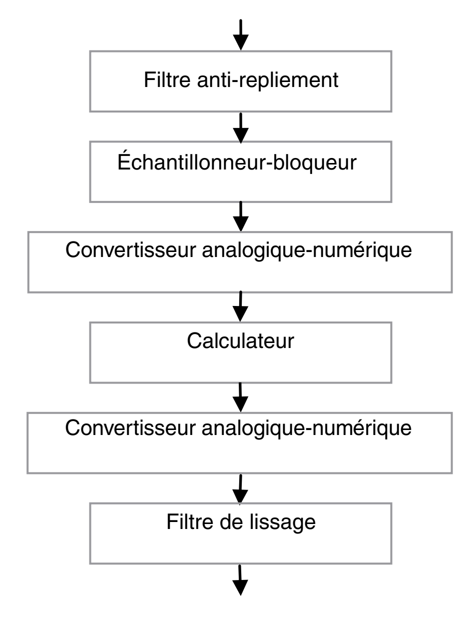


Figure 21.1 Organisation d'une chaîne de traitement numérique

Les deux opérations essentielles qui apparaissent dans la numérisation d'un signal sont l'échantillonnage (effectué par l'échantillonneur-bloqueur) et la quantification (réalisée par le convertisseur analogique-numérique).



II Échantillonnage

- **Définition**

L'échantillonnage d'un signal analogique consiste à prélever ses valeurs (appelées échantillons) à des intervalles de temps réguliers avec une cadence suffisante pour ne pas perdre d'information.
L'échantillonnage est caractérisé par sa période T_e ou par sa fréquence $F_e = \dfrac{1}{T_e}$

- **Spectre d'un signal échantillonné**

Le spectre d'un signal échantillonné est constitué par la répétition périodique le long de l'axe des fréquences du spectre du signal de départ, avec une période égale à la fréquence d'échantillonnage.

Remarque Il ne faut pas s'étonner du fait que la période soit égale à une fréquence. Le mot période est pris ici au sens mathématique et la grandeur considérée dans le spectre est une fonction de la fréquence.

- **Théorème de Shannon**

La reconstitution d'un signal à partir de ses échantillons n'est possible que si la fréquence d'échantillonnage F_e est strictement supérieure au double de la fréquence maximale du spectre du signal :

$$F_e > 2\,F_{\max}$$

Ce résultat constitue le théorème de Shannon.

Exemple La fréquence d'échantillonnage utilisée pour les cédés audio est de 44,1 kHz. Comme la bande des audiofréquences s'étend de 20 Hz à 20 kHz, le théorème de Shannon est bien respecté puisque la fréquence d'échantillonnage minimale vaut $2 \times 20 = 40$ kHz. Le choix de la valeur exacte 44,1 kHz n'est dû qu'à des raisons historiques liées aux premiers enregistrements numériques sur vidéocassettes.

La fréquence d'échantillonnage minimale, $2f_{\max}$, est appelée fréquence de Nyquist.

Remarque Quand la condition de Shannon est respectée, il n'y a pas de perte d'information lors de l'échantillonnage. Toutefois, pour retrouver le signal de départ, il faudrait disposer d'un filtre passe-bas idéal. En pra-

tique, il faut donc une fréquence d'échantillonnage un peu supérieure à la fréquence de Nyquist, avec une marge plus ou moins grande selon le filtre passe-bas utilisé.

- **Repliement**

  Si la condition de Shannon n'est pas respectée, les motifs adjacents du spectre périodique se mélangent : c'est le phénomène de repliement du spectre.

- **Filtre anti-repliement**

  Pour éviter que des composantes indésirables du signal ne provoquent un phénomène de repliement, il faut les éliminer avant l'échantillonnage. C'est le rôle du filtre anti-repliement : il s'agit d'un filtre passe-bas élimant les fréquences supérieures à $\frac{F_e}{2}$.

  La fréquence de coupure d'un filtre anti-repliement idéal doit être égale à $\frac{F_e}{2}$ En pratique, il suffit que le filtre atténue suffisamment les composantes de fréquences supérieures à $\frac{F_e}{2}$ pour que leur amplitude reste inférieure au quantum du convertisseur analogique-numérique utilisé.

III Quantification

La quantification consiste à remplacer la tension u par une tension Nq multiple d'une quantité élémentaire, le quantum q. Deux méthodes peuvent être utilisées :

– la troncature, méthode la plus simple, qui associe le nombre N à une tension u comprise entre Nq et $(N+1)q$;
– l'arrondi, méthode la plus précise, qui associe le nombre N à une tension u comprise entre $\left(N - \frac{1}{2}\right) q$ et $\left(N + \frac{1}{2}\right) q$.

La quantification conduit à une erreur d'autant plus faible que le quantum est petit.

Traitement d'un électrocardiogramme

Le système étudié est un appareil d'acquisition et de traitement d'un électrocardiogramme. La différence de potentiel qui apparaît entre des électrodes placées sur le corps humain est amplifiée par un amplificateur d'instrumentation. Le signal obtenu est ensuite numérisé.

1. Choix de la fréquence d'échantillonnage



Il s'agit de déterminer la fréquence d'échantillonnage et de mettre en évidence les précautions à prendre pour assurer une prise correcte d'échantillons portant sur la tension $v_s(t)$ issue de l'amplificateur d'instrumentation.
L'information utile du signal ECG est comprise dans la bande de fréquence $[0 ; F_{max}]$ avec $F_{max} = 100$ Hz.
a) Relever la fréquence du signal $u_{card}(t)$ de l'ECG (Fig. 21.2). Exprimer cette fréquence en battements par minute.

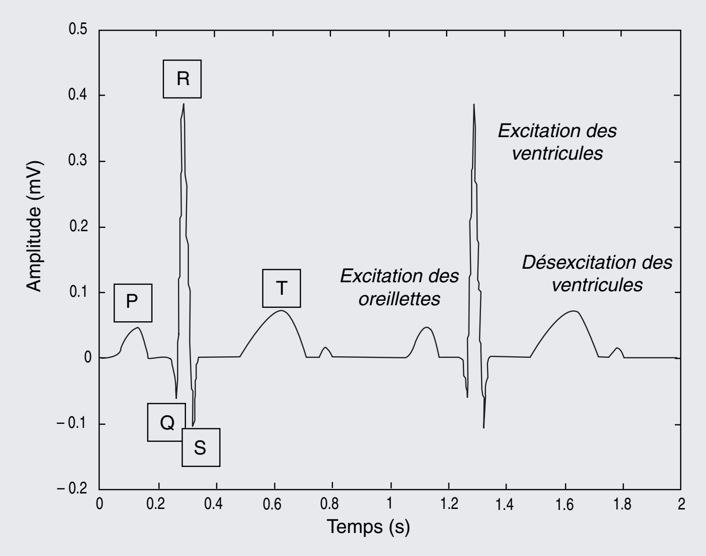


Figure 21.2 Signal ECG en fonction du temps

b) Comparer cette fréquence à celle de la première raie de fréquence non nulle du spectre de $v_s(t)$ (Fig. 21.3). Quel nom particulier donne-t-on à cette composante ?

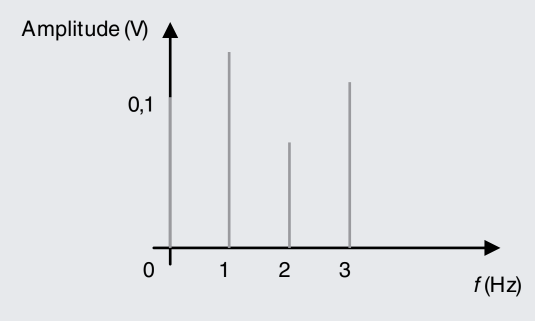


Figure 21.3 Spectre d'amplitude du signal ECG amplifié $v_s(t)$ (détail de 0 à 3 Hz)

c) D'après la condition de Shannon, à partir de quelle fréquence F_{emin} est-il possible d'échantillonner $v_s(t)$ sans altérer le signal ECG ?

F_{emin} est la fréquence minimale à partir de laquelle l'échantillonnage est correct. Pour des raisons technologiques, la fréquence d'échantillonnage retenue est F_e = 448 Hz.

2. Conséquence d'une prise d'échantillons sans filtre anti-repliement

On considère deux composantes parasites issues de la tension secteur, de fréquence f_1 = 150 Hz et f_2 = 400 Hz. On donne le spectre avant (Fig. 21.4) et après échantillonnage (Fig. 21.5).

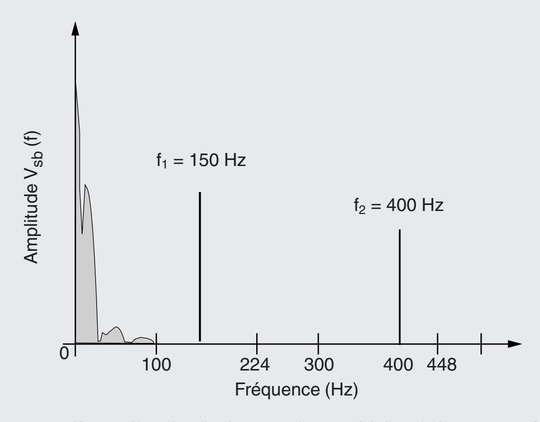


Figure 21.4 Spectre d'amplitude de la tension $v_s(t)$ bruitée avant échantillonnage

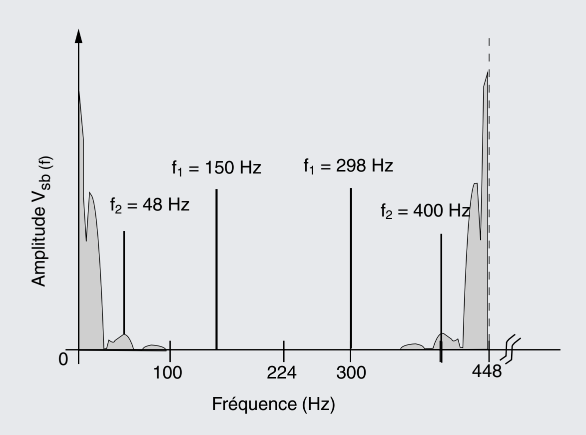


Figure 21.5 Spectre d'amplitude de la tension $v_s(t)$ bruitée après échantillonnage

a) Les fréquences f_1 et f_2 vérifient-elles la condition de Shannon ?

b) Justifier la présence de raies aux fréquences $f_1' = 298$ Hz et $f_2' = 48$ Hz.



3. Filtre anti-repliement

a) Pour une fréquence d'échantillonnage de 448 Hz, montrer que la fréquence minimale susceptible de se replier sur le spectre utile de $v_s(t)$ vaut 348 Hz.

b) Pour la fréquence d'échantillonnage choisie, quelle doit être la valeur de la fréquence de coupure du filtre anti-repliement ?

La bande passante du filtre anti-repliement est-elle compatible avec l'occupation spectrale du signal ECG ?

Solution

1.a) La période du signal $u_{card}(t)$ étant T_{card} = 1 s, sa fréquence est f_{card} = 1 Hz, ce qui correspond à 60 battements par minute.

1.b) Cette fréquence est identique à celle de la première raie du spectre : 1 Hz. Celle-ci correspond au fondamental du signal.

1.c) La condition de Shannon impose une fréquence d'échantillonnage minimale $F_{emin} = 2\ F_{\max}$, ce qui donne : $F_{emin} = 2 \times 100 = 200$ Hz.

2.a) Pour respecter la condition de Shannon, les fréquences doivent être inférieures à $\frac{F_e}{2} = 224$ Hz. C'est le cas pour la première fréquence, $f_1 = 150$ Hz, mais pas pour la seconde, $f_2 = 400$ Hz.

2.b) Le phénomène de repliement du spectre fait apparaître une composante de fréquence $f' = F_e - f$ pour chaque fréquence f. Pour les deux composantes parasites de fréquence $f_1 = 150$ Hz et $f_2 = 400$ Hz, cela donne $f'_1 = 448 - 150 = 298$ Hz et $f'_2 = 448 - 400 = 48$ Hz.

3.a) La fréquence minimale f_m susceptible de se replier dans le spectre utile est telle que $F_e - f_m = F_{\max}$, ce qui conduit à : $f_m = F_e - F_{\max}$, soit : $f_m = 448 - 100 = 348$ Hz.

3.b) La fréquence de coupure du filtre anti-repliement est $f_c = \frac{F_e}{2}$, soit : $f_c = \frac{448}{2} = 224$ Hz. La bande passante du filtre anti-repliement est compatible avec l'occupation spectrale du signal ECG puisque $f_c > F_{\max}$.

FICHE 22

Convertisseur analogique-numérique

I Présentation

- **Définition**

Un convertisseur analogique-numérique (Fig. 22.1) est un dispositif qui transforme un signal analogique u en un mot numérique de n bits, a_1, a_2,... a_n.

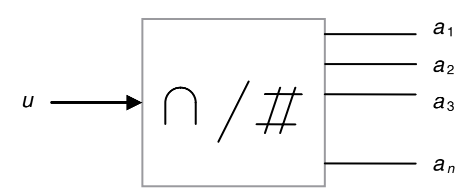


Figure 22.1 Symbole d'un convertisseur analogique-numérique

Le circuit réalise une quantification et un codage. La quantification consiste à remplacer la tension u par une tension Nq multiple d'une quantité élémentaire, le quantum q, fixé par une tension de référence V_r :

$$q = \frac{V_r}{2^n}$$

Le nombre N de quantums correspondant à la tension u est tel que :

$$|u - Nq| \leqslant \frac{q}{2}$$

Le codage consiste à convertir la valeur décimale N en un mot binaire de n bits, a_1, a_2,... a_n.

- **Caractéristique de transfert**

La caractéristique de transfert d'un convertisseur analogique-numérique donne les mots binaires de sortie en fonction de la tension appliquée à l'entrée (Fig. 22.2).

Nous avons considéré pour le tracé un convertisseur analogique-numérique 3 bits afin de bien visualiser les différents paliers.

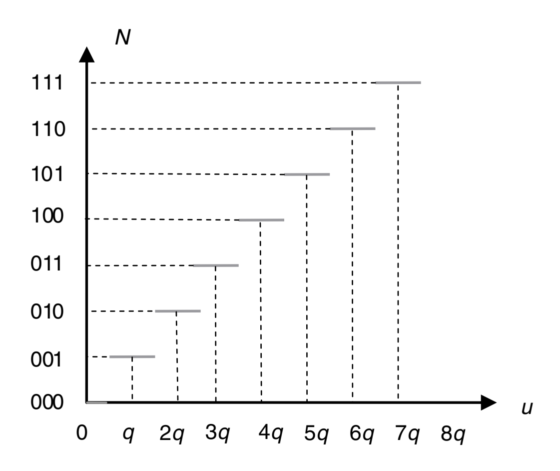


Figure 22.2 Caractéristique de transfert d'un convertisseur analogique-numérique

- **Bruit de quantification**

La différence entre la tension u à convertir et la valeur approchée Nq est l'erreur de quantification. Celle-ci est comprise entre $-\frac{q}{2}$ et $\frac{q}{2}$. Tout se passe comme si la tension u était superposée à une tension parasite appelée bruit de quantification.

- **Convertisseur unipolaire et convertisseur bipolaire**

Les convertisseurs que nous avons étudiés jusqu'à présent sont qualifiés d'unipolaires car la tension d'entrée est positive. Il existe également des convertisseurs bipolaires pour lesquels la tension d'entrée peut prendre des valeurs positives et négatives. La sortie utilise soit un code binaire décalé, soit un code complément à 2.

II Paramètres caractéristiques

- **Pleine échelle**

C'est l'étendue U_p des tensions qui peuvent être converties. Elle est fixée par la tension de référence. L'intervalle des valeurs possibles est $[0, U_p]$ pour un convertisseur unipolaire et $\left[-\frac{U_p}{2}, \frac{U_p}{2}\right]$ pour un convertisseur bipolaire.

- **Résolution**

C'est la variation minimale de tension d'entrée qui provoque un changement de mot numérique en sortie (c'est-à-dire le quantum), rapportée à la pleine échelle. Elle s'exprime en pourcentage et ne dépend que du nombre de bits du convertisseur. Par abus de langage, on caractérise souvent la résolution par le nombre de bits.

- **Durée de conversion**

C'est la durée nécessaire pour obtenir un mot numérique en sortie qui correspond au signal analogique d'entrée. Pour connaître la fréquence maximale de conversion possible, il faut calculer la période minimale en ajoutant à la durée de conversion le temps nécessaire à la remise à zéro du convertisseur.

III Principaux types

- **Convertisseur à rampe**

La tension à convertir est transformée en durée grâce à la génération d'une ou plusieurs rampes. Un compteur permet ensuite de convertir le temps en un nombre. Ce principe, utilisé dans certains voltmètres numériques, conduit à des durées de conversion élevées. Les convertisseurs à rampe sont d'un coût modeste.

- **Convertisseur à approximations successives**

Il compare la tension à convertir à des tensions de référence successives. Ce principe concilie une bonne précision et une rapidité correcte. Il est très utilisé comme convertisseur analogique-numérique associé à un capteur.

- **Convertisseur parallèle (ou flash)**

Il compare simultanément la tension à convertir à une série de seuils. L'avantage est la grande rapidité, l'inconvénient étant une plus grande complexité qui entraîne un prix plus élevé.

Banc de test pour lunettes

Un banc de test pour lunettes est un dispositif d'aide et de conseil pour la vente de lunettes de soleil (Fig. 22.3).
Après avoir disposé la paire de lunettes sur le banc, le cycle de test peut commencer. Sept sources lumineuses allant de l'ultraviolet à l'infrarouge éclairent successive-



ment les lunettes. Après détection et traitement du rayonnement transmis, un écran indique le coefficient de transmission dans chaque bande de longueur d'onde et donne un conseil d'utilisation pour les lunettes considérées.

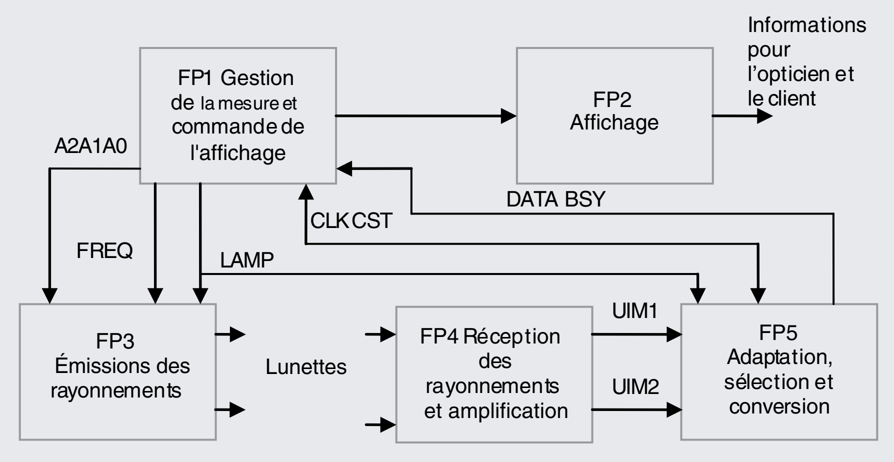


Figure 22.3 Schéma fonctionnel simplifié

L'étude porte sur la fonction FP5, « Adaptation, sélection et conversion ». La tension U_{IM1} est filtrée par un filtre sélectif centré sur 1,6 kHz. Un circuit spécialisé calcule sa valeur efficace U_{AMP1}. La tension U_{IM2} est simplement amplifiée pour donner U_{AMP2}. Le signal LAMP permet de sélectionner U_{AMP1} ou U_{AMP2} pour les convertir en binaire en vue du traitement par FP1.

Les questions portent sur le traitement de U_{IM2} (Fig. 22.4). Un extrait de la documentation du circuit intégré AD7896 est fourni en annexe.

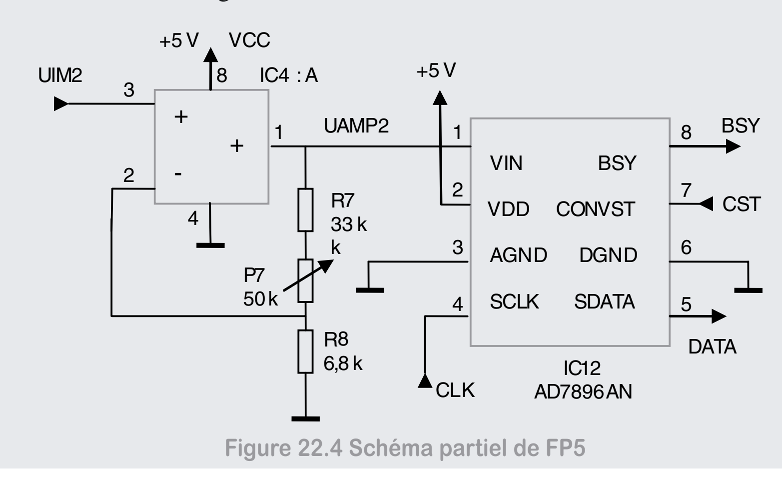


Figure 22.4 Schéma partiel de FP5

1. Donner l'expression littérale de U_{AMP2} en fonction de U_{IM2} et des éléments du montage. On donne U_{IM2} = 200 mV. À quelle valeur doit-on régler P7 pour obtenir U_{AMP2} = 2 V ?

2. Le circuit IC12 est un convertisseur analogique-numérique à sortie série. Il fournit à FP1 un mot binaire « DATA » image de la lumière transmise par les lunettes.

a) Quelles sont les limites autorisées pour U_{AMP2} ?

b) Quelle est la plus petite tension détectable par IC12 ? Donner le résultat avec quatre chiffres significatifs.

3. Déterminer le mot binaire DATA correspondant à une tension U_{AMP2} = 0,2 V.

4. On veut acquérir une donnée toutes les 100 s. Calculer la fréquence du signal CLK pour ce fonctionnement à partir de la documentation du circuit AD7896, en particulier les chronogrammes fournis (Fig. 22.5).

Annexe : documentation du circuit intégré AD7896

Convertisseur analogique-numérique 12 bits à sortie série

Temps de conversion : 8 s ; Tension d'alimentation V_{DD} : 2,7 V à 5 V

Tension d'entrée : 0 à V_{DD} ; Puissance dissipée faible : 9 mW

Numéro de broche	Symbole	Description
1	VIN	Entrée analogique, de 0 à V_{DD}
2	VDD	Tension d'alimentation de 2,7 V à 5 V
3	AGND	Masse analogique
4	SCLK	Entrée horloge série. Une horloge doit être appliquée à cette entrée pour obtenir une donnée en sortie. Quand la conversion est terminée (8 s après le front descendant de CONVST), il faut appliquer 16 périodes d'horloge sur SCLK pour obtenir la donnée SDATA.
5	SDATA	Sortie de s données en série. C'est un nombre de 12 bits émis en série. Le poids le plus fort (DB11) est émis en premier, suivi de DB10, et jusqu'au poids le plus faible DB0. En fait, 16 bits sont émis : d'abord 4 zéros puis les 12 bits du nombre. L'opération de lecture doit prendre foin au minimum 400 ns avant une nouvelle conversion.
6	DGND	Masse logique
7	CONVST	Début de conversion
8	BUSY	Sortie permettant de savoir si une conversion est en cours

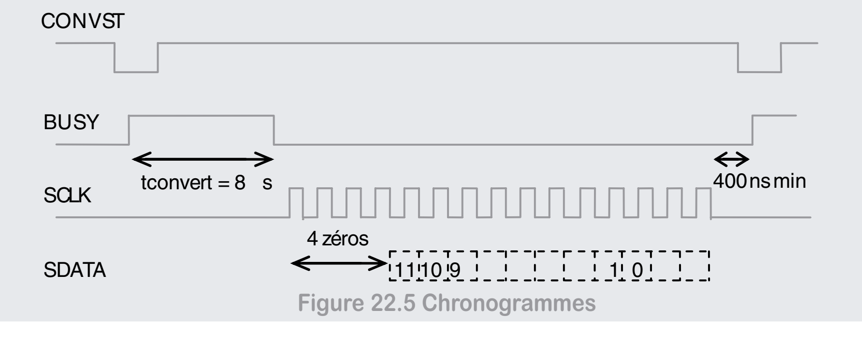


Figure 22.5 Chronogrammes



Solution

1. Le montage formé par IC4:A, R7, R8 et P7 est un amplificateur non inverseur. Son amplification est définie par $A = \dfrac{U_{AMP2}}{U_{IM2}}$ et s'exprime en fonction des éléments du montage par : $A = 1 + \dfrac{R_7 + P_7}{R_8}$. Nous en déduisons l'expression littérale de U_{AMP2} en fonction de U_{IM2} et des éléments du montage : $U_{AMP2} = \left(1 + \dfrac{R_7 + P_7}{R8}\right) U_{IM2}$. La valeur à laquelle il faut régler le potentiomètre P7 est alors :

$$P_7 = \left(\frac{U_{AMP2}}{U_{IM2}} - 1\right) R_8 - R_7$$

soit numériquement : $P_7 = \left(\dfrac{2}{0{,}2} - 1\right) \times 6{,}8 - 33 = 28{,}2\ \text{k}\Omega$.

2.a) La tension U_{AMP2} est appliquée à l'entrée du convertisseur analogique-numérique. Sa valeur doit donc être comprise entre 0 et $V_{DD} = 5$ V.

2.b) La plus petite tension détectable par le convertisseur analogique-numérique est le quantum qui s'exprime par : $q = \dfrac{V_r}{2^n}$. Le circuit intégré AD7896 étant un convertisseur 12 bits, nous obtenons : $q = \dfrac{5}{2^{12}} = 1{,}221$ mV.

3. La valeur décimale N de la grandeur de sortie correspondant à une tension d'entrée U_{AMP2} est : $N = \dfrac{U_{AMP2}}{q}$, soit : $N = \dfrac{200}{1{,}221} = 164$. La conversion en binaire donne : 10100100.

4. Les chronogrammes fournis montrent que $16T_{CLK} = 100 - 8 - 0{,}4 = 91{,}6$ s donc : $T_{CLK} = 5{,}1$ s et $f_{CLK} = 1\ 175$ kHz.

FICHE 23

Convertisseur numérique-analogique

I Présentation

- **Définition**

Un convertisseur numérique-analogique (Fig. 23.1) est un dispositif qui transforme une information numérique codée sur n bits, a_1, a_2,... a_n, en un signal analogique u (en général une tension) :

$$u = \left(\frac{a_1}{2} + \frac{a_2}{4} + \ldots + \frac{a_n}{2^n}\right) V_r$$

V_r est une tension de référence (interne ou appliquée au convertisseur) qui fixe l'échelle de la tension de sortie.

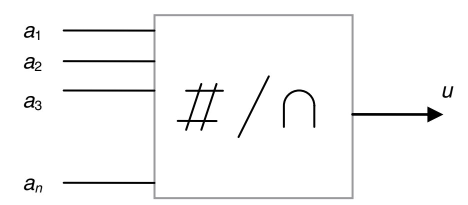


Figure 23.1 Symbole d'un convertisseur numérique-analogique

La plus petite variation de tension de sortie provoquée par un changement de mot binaire à l'entrée est le quantum :

$$q = \frac{V_r}{2^n}$$

La tension de sortie du convertisseur peut aussi s'exprimer avec le quantum :

$$u = (a_1 2^{n-1} + a_2 2^{n-2} + \ldots + a_n)q$$

- **Caractéristique de transfert**

La caractéristique de transfert d'un convertisseur numérique-analogique donne la tension de sortie en fonction des mots binaires placés à l'entrée (Fig. 23.2). Nous

avons considéré pour le tracé un convertisseur numérique-analogique 3 bits afin de bien visualiser les points.

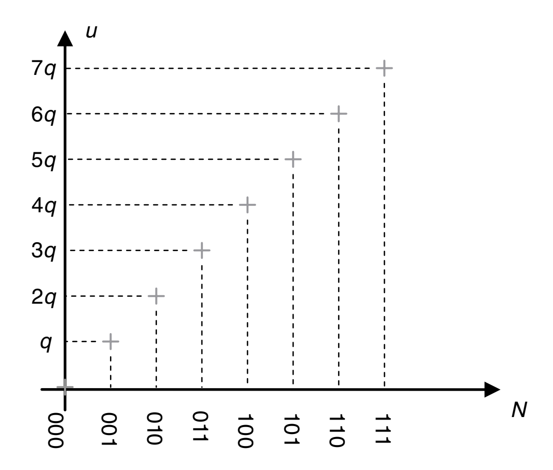


Figure 23.2 Caractéristique de transfert d'un convertisseur numérique-analogique

- **Convertisseur unipolaire et convertisseur bipolaire**

Les convertisseurs que nous avons étudiés jusqu'à présent sont qualifiés d'unipolaires car la tension de sortie est positive. Il existe également des convertisseurs bipolaires pour lesquels la tension de sortie prend des valeurs positives et négatives.

II Paramètres caractéristiques

- **Pleine échelle**

Pour un convertisseur unipolaire, la valeur maximale U_p de la tension de sortie est appelée pleine échelle. Elle est atteinte quand tous les bits sont égaux à 1 :

$$U_p = \left(1 - \frac{1}{2^n}\right) V_r = V_r - q$$

Les convertisseurs ayant en général au moins 8 bits, la pleine échelle U_p est pratiquement égale à la tension de référence V_r :

$$U_p \approx V_r$$

Dans le cas d'un convertisseur bipolaire, la pleine échelle est la différence entre le maximum et le minimum de la tension de sortie.

- **Résolution**

C'est la variation minimale de tension de sortie que l'on peut obtenir compte tenu du nombre de bits imposé à l'entrée, rapportée à la pleine échelle. Elle s'exprime en pourcentage et ne dépend que du nombre de bits du convertisseur :

$$r = \frac{q}{U_p} = \frac{1}{2^n - 1}$$

Quand on peut confondre la pleine échelle U_p avec la tension de référence V_r, la formule se simplifie :

$$r \approx \frac{1}{2^n}$$

- **Durée de conversion**

C'est la durée nécessaire pour que la tension de sortie atteigne la pleine échelle à un demi-quantum près, en partant de 0.

- **Précision**

C'est la l'écart maximal entre la valeur réelle de la tension de sortie et sa valeur théorique, exprimée en pourcentage de la pleine échelle.

III Principaux types

- **Convertisseur à résistances pondérées**

Il utilise autant de branches que de bits, chacune étant constituée d'un interrupteur commandé (transistor en commutation) et d'une résistance pondérée permettant de fixer un courant dépendant du poids de chaque bit. La somme de ces courants est ensuite convertie en tension.
Le principe du convertisseur numérique-analogique à résistances pondérées est simple, mais sa réalisation est délicate car il faut disposer de résistances de grande précision dans une large gamme de valeurs. Ce n'est possible que pour un petit nombre de bits.

- **Convertisseur à réseau R-2R**

Il utilise pour chaque bit des cellules élémentaires toutes identiques, formée de deux résistances dont les valeurs sont dans un rapport 2 et d'un interrupteur commandé. Plusieurs variantes existent (convertisseurs en échelle, en échelle inversée ou à sources de courant).



Le principe du convertisseur numérique-analogique à réseau R-2R est un peu plus compliqué, mais sa réalisation ne pose pas de problème car il suffit de disposer de résistances de grande précision dont les valeurs sont seulement dans un rapport 2, ce qui est réalisé facilement dans les circuits intégrés. La plupart des convertisseurs numérique-analogique disponibles font appel à cette technique.

- **Convertisseur à condensateurs commutés**

  Il utilise la commutation de condensateurs à la place des résistances. Les échanges de charges obtenus conduisent à un résultat similaire.

Convertisseur 8 bits

On considère un convertisseur numérique analogique 8 bits dont la tension de référence est $V_r = 5$ V.
1. Quel est le nombre de mots binaires différents qui peuvent être convertis ? Préciser l'étendue des variations du nombre décimal correspondant à l'entrée.
2. Calculer le quantum q.
3. Calculer la pleine échelle U_p.
4. En déduire la résolution r (exprimée en pourcentage).
5. Donner la valeur de la tension de sortie qui correspond à l'entrée 00100110
6. Tracer la caractéristique de transfert. Que peut-on en dire ?
7. Tracer les 8 premiers points de cette caractéristique.

Solution

1. Le nombre de mots binaires différents est 2^n, c'est-à-dire $2^8 = 256$ pour un convertisseur 8 bits. Le nombre décimal correspondant à l'entrée peut aller de 0 à 255.

2. Le quantum est donné par la formule $q = \dfrac{V_r}{2^n}$, ce qui donne pour ce convertisseur :

$q = \dfrac{5}{2^8} = 19{,}5\ \text{mV}.$

3. La pleine échelle est $U_p = \left(1 - \dfrac{1}{2^n}\right) V_r$, soit ici : $U_p = \left(1 - \dfrac{1}{2^8}\right) \times 5 = 4{,}98\ V$.

Comme nous l'avons signalé dans les rappels, la pleine échelle est très proche de la tension de référence lorsque le convertisseur possède au moins 8 bits.

4. La résolution est définie par $r = \dfrac{q}{U_p}$, ce qui donne : $r = \dfrac{0{,}0195}{4{,}98} = 0{,}392\ \%$. Un calcul direct avec la formule approchée conduit pratiquement au même résultat : $r \approx \dfrac{1}{2^8} = 0{,}390\ \%$.

5. En utilisant la formule $u = \left(\dfrac{a_1}{2} + \dfrac{a_2}{4} + \ldots + \dfrac{a_n}{2^n}\right) V_r$, la valeur de la tension de sortie qui correspond à l'entrée 00100110 est :

$$u = \left(\frac{0}{2} + \frac{0}{4} + \frac{1}{8} + \frac{0}{16} + \frac{0}{32} + \frac{1}{64} + \frac{1}{128} + \frac{0}{256}\right) \times 5 = 0{,}742 \text{ V}$$

6. La caractéristique de transfert est tracée avec la formule $u = \left(\frac{a_1}{2} + \frac{a_2}{4} + \ldots + \frac{a_n}{2^n}\right) V_r$ (Fig. 23.3). On pourrait penser que cette caractéristique est un segment de droite, mais ce n'est pas le cas, il s'agit de 256 points distincts. Toutefois, ces points sont trop rapprochés pour que l'on puisse les distinguer sur cette représentation.

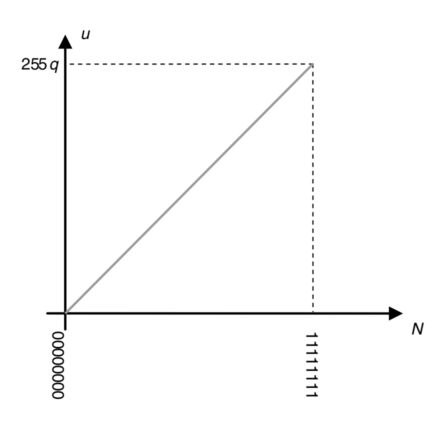


Figure 23.3 Caractéristique de transfert (vue globale)

7. En ne faisant apparaître que les 8 premiers points, on constate que la caractéristique n'est pas un segment de droite, mais qu'elle est bien formée de points distincts (Fig. 23.4).

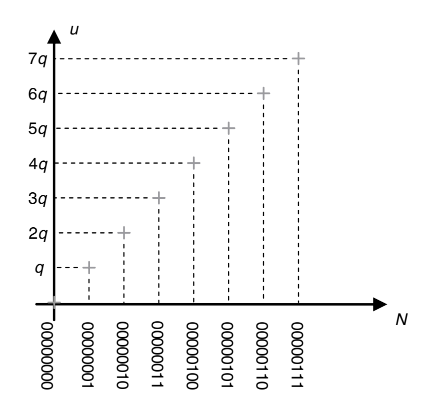


Figure 23.4 Caractéristique de transfert (premiers points)

Filtres numériques

FICHE 24

I Présentation

- **Définition**

Un filtre numérique est un système utilisé pour modifier le spectre d'un signal numérique.

- **Équation aux différences**

Elle relie le signal de sortie (y_n) au signal d'entrée (x_n). Sa forme générale est :

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots - (b_1 y_{n-1} + b_2 y_{n-2} + \ldots)$$

- **Transmittance**

La transmittance $H(z)$ d'un filtre numérique est le rapport de la transformée en z du signal de sortie et de la transformée en z du signal d'entrée :

$$H(z) = \frac{Y(z)}{X(z)}$$

Méthode Pour obtenir la transmittance en z d'un filtre numérique, il suffit de prendre la transformée en z de l'équation aux différences en utilisant les deux propriétés suivantes :

– la linéarité : $Z(\alpha x_n + \beta y_n) = \alpha Z(x_n) + \beta Z(y_n)$

– le théorème du retard qui affirme que pour tout signal numérique causal (c'est-à-dire tel que $s_n = 0$ pour $n < 0$) :

$$Z(s_{n-k}) = z^{-k} Z(s_n)$$

- **Structure**

La réalisation d'un filtre numérique est décrite par sa structure : c'est un schéma qui représente l'équation aux différences à partir de trois éléments de base (Fig. 24.1) : l'opérateur de décalage, l'additionneur-soustracteur (qui peut posséder un

nombre quelconque d'entrées) et l'opérateur de multiplication par une constante. Si les entrées de l'additionneur-soustracteur ne comportent pas de signes + et –, il est sous-entendu que l'opération à effectuer est une addition. Il existe d'autres variantes pour ces symboles.

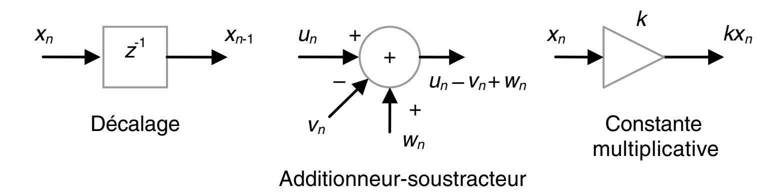


Figure 24.1 Éléments de base d'une structure

- **Stabilité**

Un filtre numérique est stable si sa réponse impulsionnelle converge vers 0. Un critère permet de savoir si un système est stable à partir de sa transmittance : il faut que tous ses pôles aient un module strictement inférieur à 1.

II Filtres numériques non récursifs

- **Définition**

La valeur de la sortie, y_n, ne dépend que des valeurs de l'entrée, x_n, x_{n-1}, x_{n-2}, ..., c'est-à-dire que l'équation aux différences est de la forme :

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots$$

- **Propriétés**

La réponse impusionnelle d'un filtre non récursif est composée d'un nombre fini d'échantillons. On qualifie ainsi ce type de filtre de RIF (réponse impulsionnelle finie). Du fait de cette propriété, un filtre récursif est toujours stable.

III Filtres numériques récursifs

- **Définition**

La valeur de la sortie, y_n, dépend des valeurs de l'entrée, x_n, x_{n-1}, x_{n-2}, ... et des valeurs précédentes de la sortie, y_{n-1}, y_{n-2}, ..., c'est-à-dire que l'équation aux différences est de la forme :

$$y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots - (b_1 y_{n-1} + b_2 y_{n-2} + \ldots)$$



- **Propriétés**

La réponse impulsionnelle d'un filtre récursif est composée d'une infinité d'échantillons. On qualifie ce type de filtre de RII (réponse impulsionnelle infinie). Un filtre récursif n'est pas toujours stable. Il faut donc appliquer le critère de stabilité donné précédemment.

Moyenneur numérique

L'exercice porte sur le traitement du signal issu d'un capteur de niveau de carburant dans le réservoir d'un avion. Le capteur a été étudié dans l'ouvrage *Électronique analogique* de la collection *Express BTS*.

Le réservoir en mouvement autour d'une position moyenne que l'on considère comme horizontale engendre sur le capteur des variations de niveau qu'il faut éliminer par traitement numérique.

Ce traitement se fait en deux étapes :

– moyennage avec sous-échantillonnage par 100, c'est à dire que l'on conserve une valeur moyennée pour 100 valeurs acquises ; cette partie du traitement permet d'éliminer les variations rapides du signal ;

– moyennage sur 16 échantillons résultats de la première étape ; cette partie assure l'élimination des variations plus lentes du signal dues au mouvement du réservoir.

Nous n'étudierons que la seconde partie du traitement numérique.

1. Étude temporelle du filtre numérique

Pour mettre en évidence l'effet de moyennage, nous étudierons un algorithme simplifié de moyenne sur 4 échantillons :

$$y_n = \frac{1}{4}(x_n + x_{n-1} + x_{n-2} + x_{n-3})$$

où x_{n-m} représente l'échantillon d'entrée retardé de m périodes d'échantillonnage et y_n l'échantillon de sortie à l'instant nT_e.

a) Représenter une structure de réalisation de cet algorithme avec les opérateurs élémentaires : addition ou soustraction, multiplication par une constante et mémorisation (retard de T_e).

b) De quel type de filtre numérique s'agit-il et quelle est sa propriété fondamentale relative à la stabilité ?

c) Pour déterminer la réponse à un échelon d'entrée ($x_n = 1$ si $n \geqslant 0$, sinon $x_n = 0$), calculer y_n pour $-1 \leqslant n \leqslant 5$ et tracer la courbe y_n en fonction de n.

d) Déduire de l'allure de la courbe, en justifiant votre réponse, la nature du filtrage réalisé.

e) Une variation non significative du niveau de liquide dans le réservoir peut être assimilée à une entrée impulsionnelle : $x_n = 1$ pour $n = 0$, sinon $x_n = 0$. Calculer y_n pour $-1 \leqslant n \leqslant 5$ et tracer la courbe y_n en fonction de n.

f) Justifier l'intérêt de ce type d'algorithme par rapport à l'objectif désiré.

2. Étude fréquentielle du filtre numérique

a) Établir la fonction de transfert en z de ce filtre : $T(z) = \dfrac{Y(z)}{X(z)}$.

b) En effectuant le changement de variable $z = e^{j\omega T_e}$ avec ω pulsation du signal d'entrée et T_e période d'échantillonnage, établir la fonction de transfert isochrone $\underline{T}$.

c) $\underline{T}$ peut s'écrire $\underline{T} = 0{,}5[\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e)]e^{-j1{,}5\omega T_e}$. En déduire le module de la fonction de transfert $|\underline{T}|$.

d) Montrer que l'argument de la transmittance de ce filtre, $\varphi = \arg \underline{T}$, s'exprime par $\varphi = -1{,}5\omega T_e$ dans le domaine de fréquences tel que $\dfrac{f}{F_e} < \dfrac{1}{4}$ pour lequel $\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e) > 0$.

e) En déduire l'expression du retard τ introduit par ce filtre dans la transmission des informations.

f) Préciser le domaine de fréquences utile en relation avec l'échantillonnage.

Solution

1.a) Une structure possible consiste à obtenir x_{n-1}, x_{n-2} et x_{n-3} par des opérateurs retard, puis en faire la somme par un opérateur additionneur et enfin, multiplier le résultat par $\dfrac{1}{4}$ avec un opérateur multiplication par une constante (Fig. 24.2). L'énoncé demande *une* structure de réalisation : il existe en effet d'autres solutions que celle qui est proposée, en effectuant les opérations dans un ordre différent.

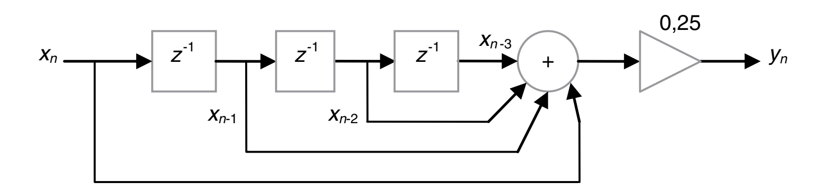


Figure 24.2 Structure de réalisation du filtre

1.b) Il s'agit d'un filtre non récursif : la valeur de la sortie y_n ne dépend que des valeurs d'entrée x_n, x_{n-1}, x_{n-2},... Un filtre non récursif est toujours stable.

1.c) Pour déterminer la réponse à un échelon, on remplace les valeurs de l'entrée dans l'équation aux différences. Les résultats sont consignés dans le tableau suivant :

n	– 1	0	1	2	3	4	5
x_n	0	1	1	1	1	1	1
y_n	0	0,25	0,5	0,75	1	1	1

Ces valeurs permettent de tracer la courbe y_n en fonction de n (Fig. 24.3).

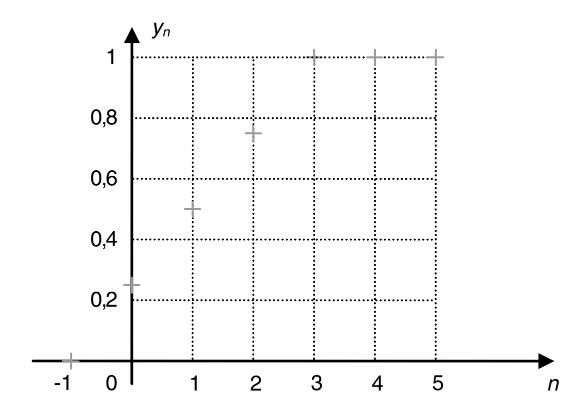


Figure 24.3 Réponse indicielle du filtre

1.d) Le filtre est passe-bas car il laisse passer le continu et sa réponse indicielle ne présente pas de front.
1.e) Pour déterminer la réponse à une impulsion, on remplace les valeurs de l'entrée dans l'équation aux différences. Les résultats sont consignés dans le tableau suivant :

n	– 1	0	1	2	3	4	5
x_n	0	1	0	0	0	0	0
y_n	0	0,25	0,25	0,25	0,25	0	0

Ces valeurs permettent de tracer la courbe y_n en fonction de n (Fig. 24.4).

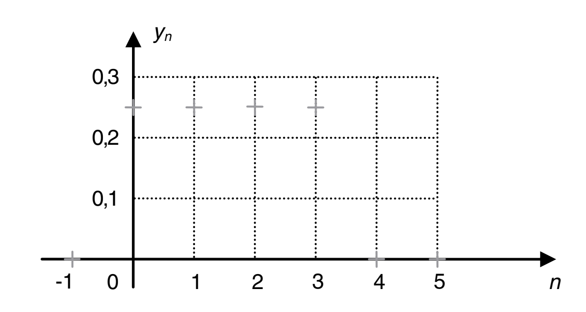


Figure 24.4 Réponse impulsionnelle du filtre

1.f) L'influence d'une perturbation est réduite d'un facteur 4.

2.a) Pour passer de l'équation aux différences à la transmittance en z, la première étape consiste à prendre la transformée en z des deux membres de l'équation en tenant compte de la linéarité de la transformation et du théorème du retard : $Y(z) = \frac{1}{4}[X(z) + z^{-1}X(z) + z^{-2}X(z) + z^{-3}X(z)]$. La seconde étape consiste à diviser les deux membres de la formule obtenue par $X(z)$ afin de faire apparaître la transmittance : $T(z) = \frac{Y(z)}{X(z)} = \frac{1}{4}(1 + z^{-1} + z^{-2} + z^{-3})$.

2.b) La transmittance isochrone s'obtient à partir de la transmittance en z en remplaçant z par $e^{j\omega T_e}$, ce qui donne : $\underline{T} = \frac{1}{4}(1 + e^{-j\omega T_e} + e^{-2j\omega T_e} + e^{-3j\omega T_e}$.

2.c) Le module d'un produit est égal au produit des modules des différents facteurs. Par ailleurs, le module d'un nombre réel est égal à sa valeur absolue et le module d'une exponentielle d'imaginaire pur est égal à 1. En appliquant ces propriétés à la formule donnée par l'énoncé, $\underline{T} = 0{,}5[\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e)]e^{-j1{,}5\omega T_e}$ (qui n'est pas à démontrer), nous obtenons : $|\underline{T}| = 0{,}5|\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e)|$.
Il ne faut pas oublier la valeur absolue car $\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e)$ n'est pas toujours positif.

2.d) L'argument d'un produit est égal à la somme des arguments des différents facteurs. Par ailleurs, l'argument d'un nombre réel positif est nul et l'argument d'une exponentielle d'imaginaire pur est égal à son exposant.
En appliquant ces propriétés à la formule donnée par l'énoncé, nous obtenons bien : $\varphi = -1{,}5\omega T_e$ puisque $\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e) > 0$ dans le domaine de fréquence considéré.

> **Remarques** • L'énoncé nous donne une forme de transmittance propice au calcul du module et de l'argument. Ce résultat aurait pu être démontré en cherchant à factoriser $e^{-j1{,}5\omega T_e}$.
> • Dans le domaine de fréquence où $\cos(1{,}5\omega T_e) + \cos(0{,}5\omega T_e) < 0$, l'argument devient $\varphi = -1{,}5\omega T_e + \pi$ car l'argument d'un nombre réel négatif vaut π.

2.e) Si un filtre introduit un retard τ, l'argument de sa transmittance est $\varphi = -\omega\tau$. Par identification avec l'argument obtenu ici, $\varphi = -1{,}5\omega T_e$, nous obtenons $\tau = 1{,}5\omega T_e$.

2.f) Le théorème de Shannon impose la condition $f < \frac{F_e}{2}$ pour obtenir un échantillonnage correct.

FICHE 25

Codage en bande de base

I Introduction

- **Organisation d'une transmission numérique**

Un signal numérique est constitué d'une suite d'éléments binaires (0 ou 1). Pour le transmettre, il faut le traduire en grandeur physique adaptée au canal de transmission : c'est le but du codage en bande de base réalisé par l'émetteur. Le récepteur permet de retrouver le signal numérique à partir de la grandeur physique transmise (Fig. 25.1). Dans une transmission en bande de base, le signal ne subit pas de transposition de fréquence.



Figure 25.1 Organisation d'une transmission numérique

Cette configuration minimale peut être complétée par un codage de source et un codage de canal. Le codeur de source, placé juste après la source de signal numérique profite des redondances souvent présentes dans le message pour réduire sa longueur. Le codeur de canal, placé ensuite, ajoute des informations permettant de détecter et corriger les erreurs de transmission. Cette opération est réalisée par le décodeur de canal qui suit le récepteur. Le décodeur de source placé ensuite permet de retrouver le signal initial.

- **Débit binaire**

L'intervalle de temps nécessaire à la transmission de chaque élément binaire a une durée T_B fixée, appelée durée d'un bit. Le débit binaire D est l'inverse de la durée d'un bit :

$$D = \frac{1}{T_B}$$

D s'exprime en bits par seconde (symbole bps).

- **Taux d'erreur binaire**

Le taux d'erreur binaire (TEB) est le rapport entre le nombre de bits reçus erronés et le nombre total de bits transmis.

- **Codage binaire et codage *M*-aire**

Le signal numérique peut être transmis bit par bit : ce type de codage est qualifié de binaire. La séquence binaire peut aussi être convertie en suite de symboles constitués chacun de n bits. Il existe alors $M = 2^n$ symboles différents : ce type de codage est qualifié de M-aire.

> Exemple Le code 2B/1Q, utilisé notamment dans les RNIS (réseaux numériques à intégration de services), emploie des symboles de 2 bits : il est donc quaternaire ($M = 4$). Aux symboles 00, 01, 11 et 10 correspondent les niveaux de tension $-3A$, $-A$, A et $3A$.

- **Rapidité de modulation**

L'intervalle de temps nécessaire à la transmission de chaque symbole a une durée T_S fixée, appelée durée d'un symbole. La rapidité de modulation est le nombre de symboles transmis par seconde et son unité est le baud (symbole Bd). C'est l'inverse de la durée d'un symbole :

$$R = \frac{1}{T_S}$$

Les unités bit par seconde et baud correspondent à des grandeurs de même dimension (l'inverse d'un temps). La distinction sert à identifier facilement la quantité considérée : débit binaire ou rapidité de modulation.

La rapidité de modulation et le débit binaire sont liés :

$$R = \frac{D}{n} = \frac{D}{\log_2 M}$$

où $\log_2$ est le logarithme de base 2.
Dans le cas d'un codage binaire, la rapidité de modulation s'exprime par le même nombre que le débit binaire (mais pas avec la même unité).

II Exemples de codes

- **Code NRZ**

Le codage NRZ (*No Return to Zero*) est le plus simple : le 0 logique est codé par un niveau de tension A (positif ou négatif) et le 1 par un niveau symétrique $-A$.

L'intérêt est de ne pas utiliser de niveau nul qui pourrait être confondu avec une absence de signal.

Le codage NRZ est utilisé par exemple pour les liaisons série RS232 et pour les bus CAN rencontrés dans le domaine de l'automobile. L'état haut ne correspond pas toujours au 1 logique. Par exemple, une liaison RS232 utilise souvent un niveau + 12 V pour le 0 logique et un niveau – 12 V pour le 1.

L'encombrement spectral est approximativement $\left[0, \frac{1}{T_b}\right]$ (pour 90 % de la puissance). Le spectre présentant un maximum à la fréquence nulle, le canal de transmission doit laisser passer le continu. Il est impossible de récupérer l'horloge car la densité spectrale s'annule à la fréquence $1/T_b$.

L'inconvénient majeur du codage NRZ réside dans la difficulté de synchronisation du récepteur quand le signal ne présente pas de transition lors d'une longue séquence de 1 ou 0. Dans certains cas, cet inconvénient peut être évité par des artifices : sur le bus CAN, on utilise la méthode du bourrage de bits (*bit stuffing*) en imposant un changement d'état au bout de cinq bits identiques consécutifs.

- **Code RZ**

Dans le codage RZ (*Return to Zero*), le 0 logique est codé par un niveau de tension nul et le 1 par une tension qui prend une valeur A pendant la moitié de la durée d'un bit et qui revient à zéro pendant l'autre moitié.

L'encombrement spectral est à peu près $\left[0, \frac{2}{T_b}\right]$ (pour 95 % de la puissance). Le signal possède une composante continue et il faut donc que le canal de transmission laisse passer le continu. Le spectre comporte une raie à la fréquence $1/T_b$, ce qui permet de récupérer le rythme de l'horloge à partir du signal transmis.

- **Code Manchester**

Le codage Manchester résout le problème des longues séquences de 1 ou 0 en imposant des transitions placées au milieu de l'intervalle élémentaire correspondant à la durée d'un bit : le 0 logique est représenté par un front montant du signal et le 1 logique correspond à un front descendant (ou parfois l'inverse).

Le codage Manchester est utilisé par exemple dans les réseaux Ethernet.

L'encombrement spectral est approximativement $\left]0, \frac{2}{T_b}\right]$. La densité spectrale s'annulant à la fréquence nulle, il est possible d'utiliser un canal de transmission qui ne laisse pas passe le continu (par exemple parce qu'il comporte des transformateurs de liaison). La récupération du rythme de l'horloge est possible.

Comparaison de deux codes en bande de base

Le débit binaire est D = 1200 bits par seconde.

1. Codage NRZ (*No Return to Zero*)

Ce codage est défini par :

– état 1 : niveau + V_D,
– état 0 : niveau – V_D, avec V_D = 2,5 V.

a) Exprimer la durée d'un bit T_B en fonction du débit binaire D.
b) $s_N(t)$ représente le signal de sortie du système de codage NRZ pour une séquence de données binaires présentes à l'entrée. Compléter la courbe de $s_{N2}(t)$ (Fig. 25.2).

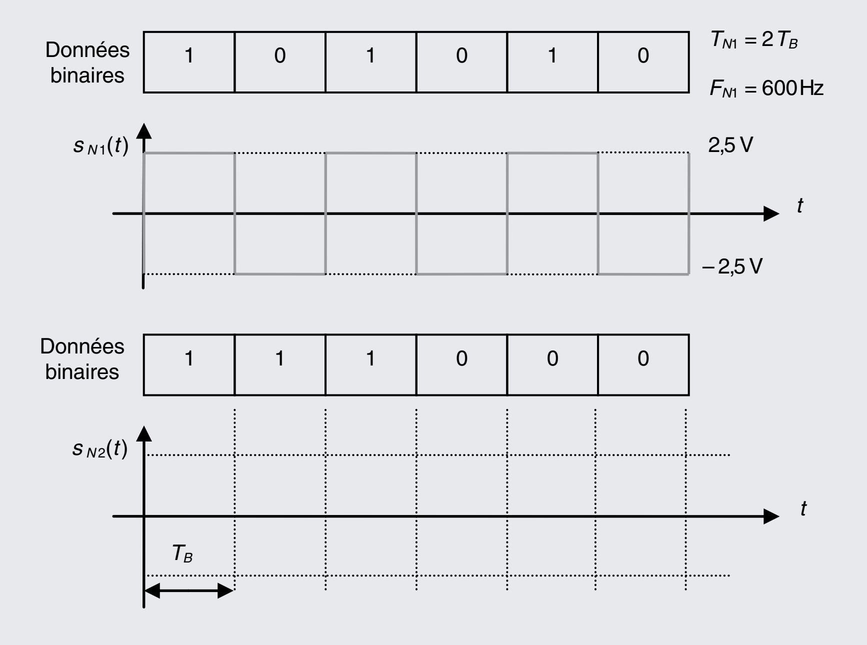


Figure 25.2 Codage NRZ

c) Sachant que les données binaires correspondent à une séquence périodique 111 000 111 000, etc., exprimer la période T_{N2} en fonction de T_B et en déduire la fréquence F_{N2}.
d) En supposant que l'on émette une très longue série de 1, quelles sont les caractéristiques du signal électrique obtenu (forme, valeur moyenne, fréquence) ?

2. Codage Manchester

Ce codage est défini par :

– état 1 : front montant pendant la durée d'un bit (T_B),

– état 0 : front descendant pendant la durée d'un bit (T_B).

a) $s_M(t)$ représente le signal de sortie du système de codage Manchester pour une séquence de données binaires présentes à l'entrée. Compléter les courbes de $s_{M1}(t)$ et $s_{M1}(t)$ (Fig. 25.3).

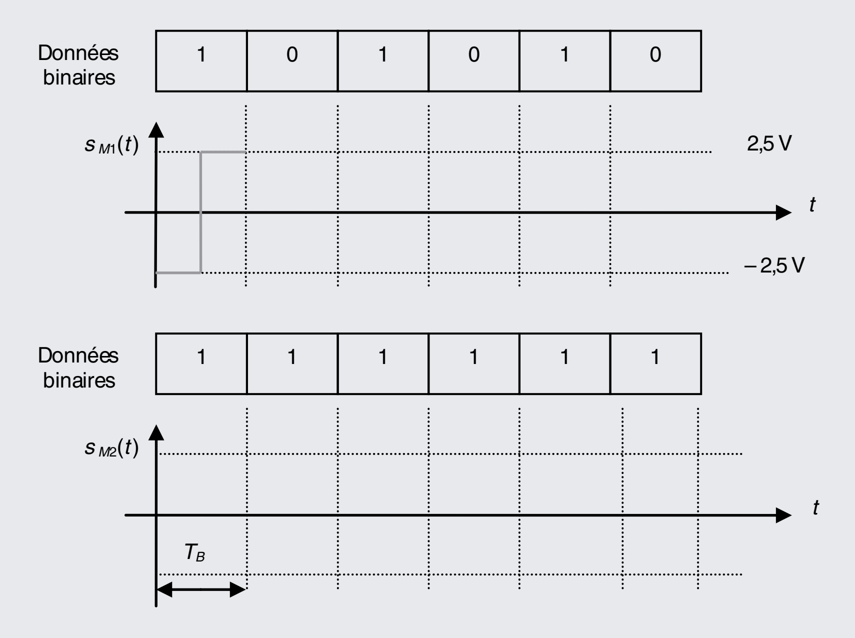


Figure 25.3 Codage Manchester

b) Exprimer les deux périodes T_{M1} et T_{M2} des signaux en fonction de T_B. En déduire les fréquences F_{M1} et F_{M2} correspondantes.

3. Spectres des codes NRZ et Manchester

On a représenté la DSP (Densité Spectrale de Puissance) relative des signaux aléatoires codés NRZ et Manchester (Fig. 25.4). L'encombrement spectral pourra être assimilé à la largeur du premier lobe de la DSP.

a) Estimer graphiquement l'encombrement spectral dans chaque cas : B_{NRZ} pour le codage NRZ et B_{MAN} pour le codage Manchester.

b) Le critère de choix pour la carte étudiée est de minimiser l'encombrement en fréquence. Quel est le codage le plus adapté ?

c) Si les critères de choix avaient été : la DSP doit être minimale en basse fréquence et avoir un certain niveau à la fréquence d'horloge (1200 Hz), quel codage serait le plus approprié ? Justifier la réponse.

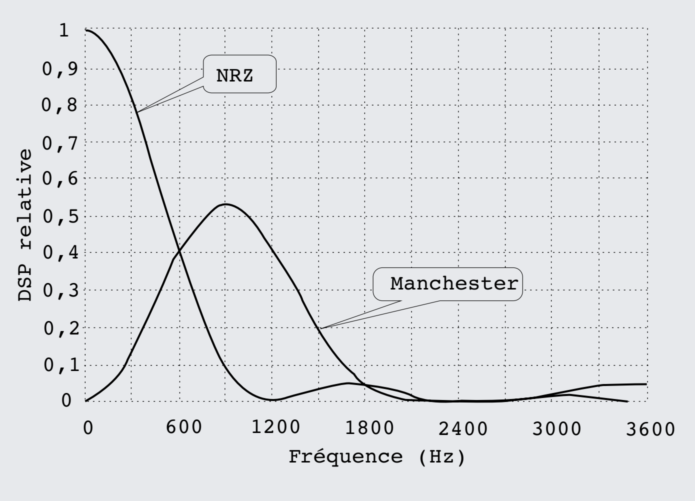


Figure 25.4 Spectres des signaux NRZ et Manchester

Solution

1.a) Le débit binaire est défini à partir de la durée d'un bit T_B par $D = \frac{1}{T_B}$, ce qui conduit à : $T_B = \frac{1}{D}$.

1.b) Le signal $s_{N2}(t)$ vaut V_D = 2,5 V quand la donnée binaire est 1 et $-V_D$ = – 2,5 V quand la donnée binaire est 0 (Fig. 25.5).

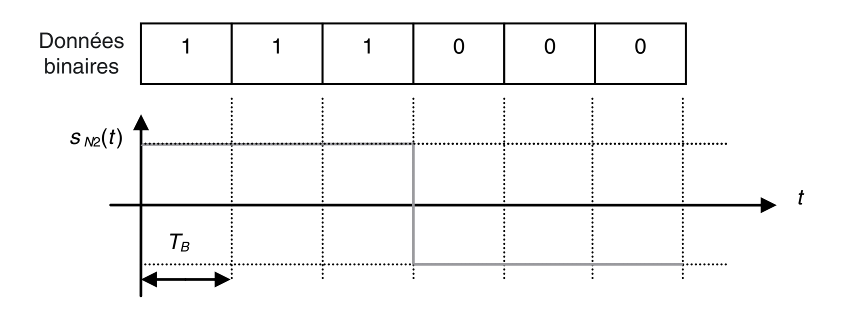


Figure 25.5 Signal codé NRZ

1.c) La séquence répétant un motif de six bits 111 000, la période est $T_{N2} = 6T_B$. La fréquence correspondante est $F_{N_2} = \dfrac{1}{T_{N_2}} = \dfrac{1}{6T_B} = \dfrac{D}{6}$, soit : $F_{N_2} = \dfrac{1\,200}{6} = 200$ Hz.

1.d) Si la séquence est une longue suite de 1, le signal $s_{N2}(t)$ reste toujours à la valeur $V_D = 2{,}5$ V. Il s'agit donc d'un signal continu de valeur moyenne 2,5 V. La fréquence n'est pas définie puisque le signal n'est pas périodique, mais dans les spectres, on place le continu à la fréquence nulle.

2.a) Le signal $s_M(t)$ présente un font montant au milieu d'un intervalle correspondant à la durée d'un bit lorsque la donnée binaire est 1 et un front descendant quand la donnée binaire est 0 (Fig. 25.6).

2.b) Pour $s_{M1}(t)$, la période est $T_{M_1} = 2T_B$ et la fréquence correspondante $F_{M_1} = \dfrac{1}{T_{M_1}} = \dfrac{1}{2T_B} = \dfrac{D}{2}$, soit $F_{M_1} = \dfrac{1\,200}{2} = 600$ Hz. Pour $s_{M2}(t)$, la période est $T_{M_2} = T_B$ et la fréquence correspondante $F_{M_2} = \dfrac{1}{T_{M_2}} = \dfrac{1}{T_B} = D$, soit $F_{M_2} = 1\,200$ Hz.

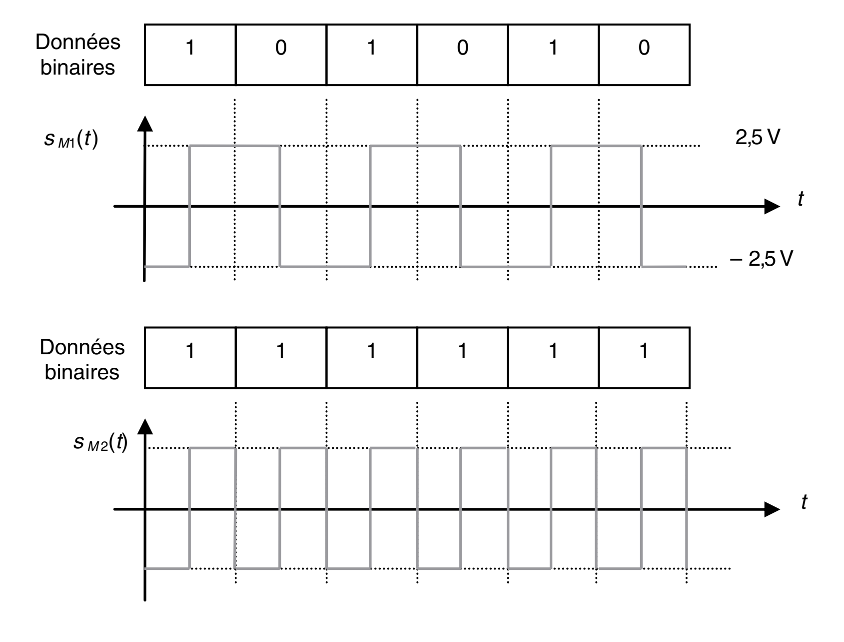


Figure 25.6 Signal codé Manchester

3.a) L'encombrement spectral correspond à la largeur du lobe principal de la courbe de la DSP, soit : B_{NRZ} = 1200 Hz pour le codage NRZ et B_{MAN} = 2 400 Hz pour le codage Manchester (Fig. 25.7).

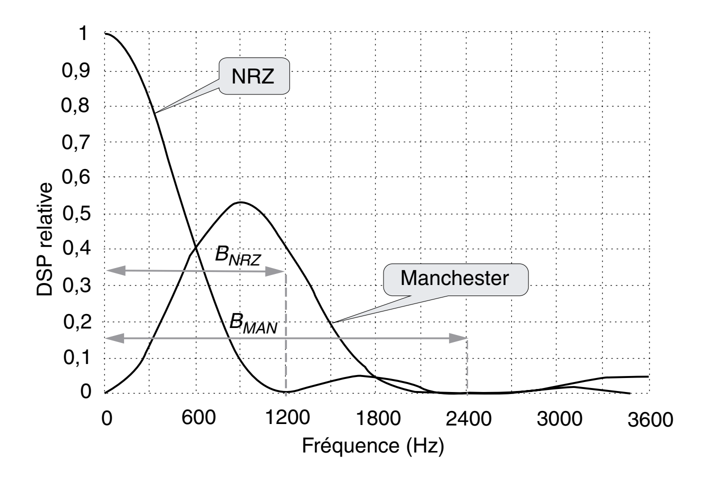


Figure 25.7 Encombrement spectral

3.b) Le codage le mieux adapté est le NRZ puisque son encombrement en fréquence (1200 Hz) est inférieur à celui du Manchester (2 400 Hz).
3.c) Le codage le mieux adapté est le Manchester car sa DSP tend vers 0 aux basses fréquences (alors qu'elle est maximale pour le codage NRZ) et qu'elle présente une valeur élevée à 1 200 Hz (alors qu'elle est nulle pour le codage NRZ).

FICHE 26

Modulations numériques

I Introduction

- **Définition**

Dans une modulation numérique, les signaux numériques modulent une porteuse sinusoïdale pour obtenir un signal adapté au canal de transmission. Pendant un intervalle élémentaire, un paramètre de la porteuse prend une valeur caractéristique du symbole à transmettre. Comme en modulation analogique, le paramètre à modifier peut être :

– l'amplitude,
– la fréquence,
– la phase.

- **Modulation binaire et modulation *M*-aire**

Le signal numérique peut être transmis bit par bit, avec un paramètre qui ne prend que deux valeurs : ce type de modulation est qualifiée de binaire. La séquence binaire peut aussi être convertie en suite de symboles constitués de n bits, avec un paramètre qui prend M valeurs. Il existe alors $M = 2^n$ symboles différents : ce type de modulation est qualifié de M-aire.

II Modulation par déplacement de fréquence

- **Définition**

Dans la modulation par déplacement de fréquence (FSK pour *Frequency Shift Keying*), les données binaires font varier la fréquence d'une porteuse sinusoïdale.

- **Modulation par déplacement de fréquence à deux états**

Pour une modulation par déplacement de fréquence à deux états (BFSK pour *Binary Frequency Shift Keying*), la fréquence du signal modulé peut prendre deux valeurs, f_0 et f_1 selon le symbole d'un bit à transmettre :

Symbole	Fréquence
0	f_0
1	f_1

Les deux fréquences sont symétriques par rapport à la fréquence de la porteuse f_p (obtenue en l'absence de signal modulant) :

$$f_0 = f_p - \Delta f$$

$$f_1 = f_p + \Delta f$$

Δf est appelé excursion en fréquence.
L'indice de modulation h est défini par le rapport de l'écart entre les deux fréquences à la rapidité de modulation R :

$$h = \frac{|f_1 - f_0|}{R}$$

Il existe deux types de modulation par déplacement de fréquence :

– la modulation à phase discontinue pour laquelle le signal présente des discontinuités à l'instant des transitions ;
– la modulation à phase continue pour laquelle le signal évolue sans discontinuité à l'instant des transitions.

La modulation à phase discontinue présente l'avantage de la simplicité, mais elle occupe une bande de fréquence plus large que la modulation à phase continue.

- **Applications**

La modulation FSK a été utilisée dans les premières générations de modems. Elle est aujourd'hui employée dans les applications de télémesure qui ne nécessitent pas de hauts débits.

- **Modulation à déplacement minimal**

La modulation à déplacement minimal (MSK pour *Minimum Shift Keying*) est une modulation FSK binaire à phase continue avec un indice de modulation h = 0,5. Son intérêt est de concentrer la densité spectrale de puissance autour de la fréquence de la porteuse. La phase varie continûment et linéairement par intervalles. Un 0 fait varier la phase de – 90° et un 1 la fait varier de + 90°.
Le téléphone mobile GSM utilise une variante de modulation MSK appelée GMSK (*Gaussian Minimum Shift Keying*) pour laquelle le signal en bande de base est filtré par un filtre passe-bas gaussien afin de réduire l'encombrement spectral du signal modulé.



III Modulation par saut de phase

- **Définition**

Dans la modulation par saut de phase (MDP ou PSK pour *Phase Shift Keying*), les données binaires font varier la phase d'une porteuse sinusoïdale.

- **Modulation par saut de phase à deux états**

Dans la modulation de phase à deux états (MDP2 ou BPSK pour *Binary Phase Shift Keying*), la phase peut prendre deux valeurs selon le symbole d'un bit à transmettre :

Symbole	Phase
0	0
1	π

On représente cela par un diagramme de constellation (Fig. 26.1). Pour cela, on place dans un plan deux points, correspondant aux symboles 0 et 1. La distance d'un point à l'origine donne l'amplitude et l'angle que fait la direction du point par rapport à l'axe des abscisses donne la phase. L'axe des abscisses est appelé I (*Inphase*, en phase) et l'axe des ordonnées est nommé Q (*Quadrature*).

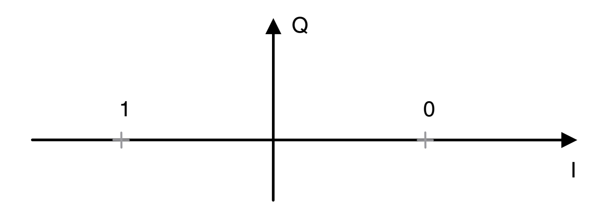


Figure 26.1 Diagramme de constellation

- **Modulation par saut de phase à quatre états**

Dans la modulation de phase à quatre états (MDP4 ou QPSK pour *Quadrature Phase Shift Keying*), la phase peut prendre quatre valeurs selon le symbole de deux bits à transmettre :

Symbole	Phase
00	0
01	$\frac{\pi}{2}$
10	π
11	$-\frac{\pi}{2}$

Le diagramme de constellation comporte quatre points (Fig. 26.2).

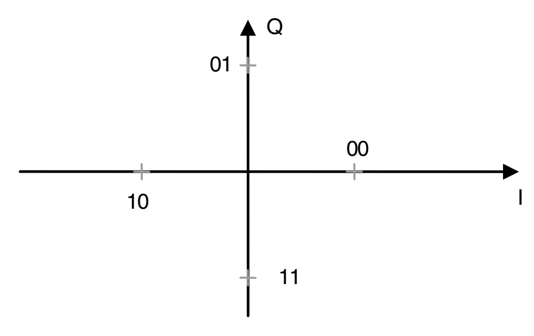


Figure 26.2 Diagramme de constellation

- **Applications**

  La modulation par saut de phase est la technique la plus utilisée. Elle est employée aussi bien pour les réseaux locaux ou les modems que pour les transmissions par satellite : télévision, systèmes de positionnement (GPS)...

Modulation FSK

Une balise maritime transmet des informations numériques au poste central en utilisant une modulation FSK.
Les signaux émis ont les caractéristiques suivantes :
– Type de modulation : BFSK (*Binary Frequency Shift Keying*)
– Fréquence de la porteuse : $F_p = 151{,}650$ MHz
– Excursion en fréquence : $\Delta F = 2{,}5$ kHz
– Largeur d'un canal : $B_{\text{canal}} = 12{,}5$ kHz
– Débit binaire : $D = 1\,200$ bps (bits par seconde)
Les porteuses adjacentes sont à 12,5 kHz de la porteuse étudiée.
L'encombrement en fréquence B_{f_1} d'un signal modulé FSK est estimé par la relation (dérivée de la règle de Carson) : $B_{f_1} = 2(R + \Delta F)$ où R est la rapidité de modulation.
La fréquence basse F_L correspond à un niveau logique 0.
La fréquence haute F_H correspond à un niveau logique 1.
1. Donner les valeurs des deux fréquences F_H et F_L.
2. En modulation BFSK, $R = D$, calculer l'encombrement en fréquence B_{f_1} du signal FSK
3. Indiquer sur un axe de fréquence gradué (Fig. 26.3) :
– les limites des canaux (barres verticales),
– les porteuses F_{pp} et F_{ps} des deux canaux adjacents,

– les deux fréquences F_H et F_L,
– l'encombrement en fréquence du signal modulé calculé précédemment (on le représentera par une zone hachurée).

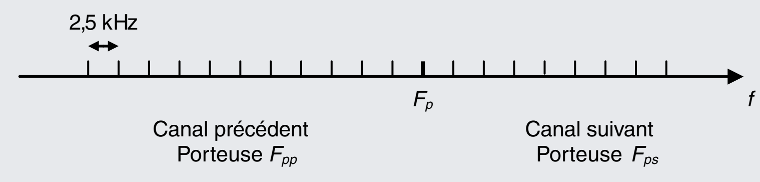


Figure 26.3 Axe des fréquences

4. Le spectre du signal modulé a été relevé (Fig. 26.4). Repérer sur cette courbe l'emplacement de la porteuse et les limites du canal (le jour de la mesure, les canaux adjacents n'étaient pas occupés).

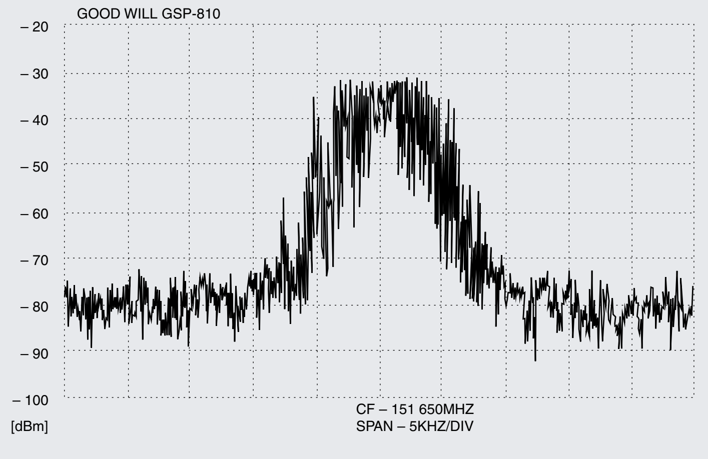


Figure 26.4 Spectre du signal modulé

Solution

1. La fréquence haute est $F_H = F_p + \Delta F$, ce qui donne numériquement : $F_H = 151{,}650 + 2{,}5 \times 10^{-3} = 151{,}6525$ MHz.
La fréquence basse est $F_L = F_p - \Delta F$, soit : $F_L = 151{,}650 - 2{,}5 \times 10^{-3} = 151{,}6475$ MHz.
Compte tenu du faible écart existant entre les différentes fréquences, il faut donner des résultats avec un nombre de chiffres significatifs suffisant.
2. Comme $R = D$ en modulation BFSK, la formule de l'encombrement spectral donnée par l'énoncé, $B_{f_1} = 2(R + \Delta F)$ devient : $B_{f_1} = 2(D + \Delta F)$. L'application numérique

conduit à : $B_{f_1} = 2(1\ 200 + 2\ 500) = 7\ 400$ Hz. Il faut remarquer que bien que D et ΔF n'aient pas la même unité, la formule est homogène : le bit par seconde ou le hertz sont tous les deux des unités de grandeurs qui correspondent à l'inverse d'un temps.

3. Les différentes grandeurs demandées par l'énoncé sont indiquées sur l'axe des fréquences (Fig. 26.5).

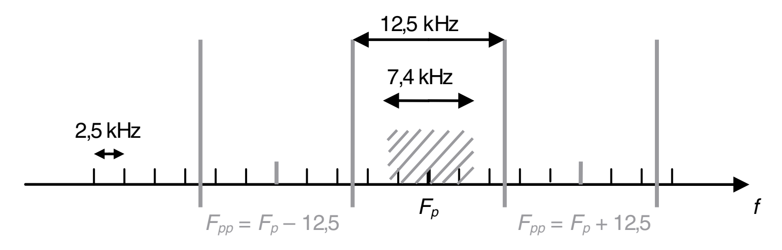


Figure 26.5 Axe des fréquences complété

4. La fréquence de la porteuse, 151,650 MHz, correspond au milieu du canal dont la largeur est de 12,5 kHz (Fig. 26.6).

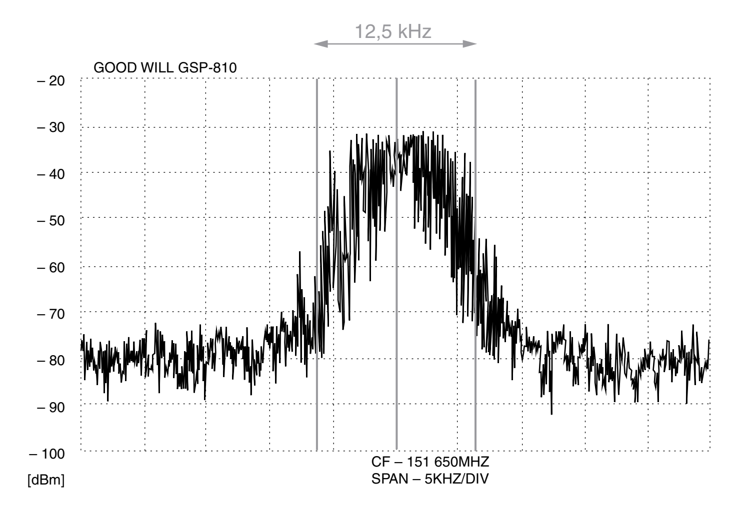


Figure 26.6 Emplacement de la porteuse et limites du canal

Index

M

N

O, P

Q, R

S

T

U, V